

Ehud Reiter

Natural Language Generation

 Springer

Natural Language Generation

Ehud Reiter

Natural Language Generation

Ehud Reiter
Department of Computer Science
University of Aberdeen
Aberdeen, UK

ISBN 978-3-031-68581-1 ISBN 978-3-031-68582-8 (eBook)
<https://doi.org/10.1007/978-3-031-68582-8>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

I have been working on natural language generation (NLG), that is using artificial intelligence techniques to produce texts in English and other human languages, since I got my PhD in this area in 1990. In late 2022, NLG became much more prominent because of the impressive capabilities of large language models such as ChatGPT, which was exciting. However, discussions about NLG in both academic and commercial circles have become focused on the latest developments in language models, with little attention paid to what had been learned about NLG before ChatGPT.

My **goal** in this book is to present a broad overview of NLG which talks about language models, but also looks at alternative approaches to NLG, requirements (what users want NLG systems to do), evaluation, safety and testing, and sensible applications of NLG. I hope that this broad perspective, which builds on decades of work on NLG, will be helpful to both researchers and developers who work in this area.

I co-authored a book about NLG in 2000, and I saw that while the technology content of my 2000 book quickly became out of date, people kept on using it in 2010 and even 2020 because the high-level conceptual and methodological material was still useful. With this experience in mind, I have focused this book on high-level concepts and methodologies; my hope is that this material will still be useful in 2030 and perhaps even 2040. I do not attempt to describe the latest technologies, because this information quickly becomes dated (indeed, anything I write in June 2024 will probably be out-of-date by the time the book is published); readers interested in the latest developments in language models should look elsewhere.

In many places I show outputs from ChatGPT and other large language models. Most of these were all produced in 2023 (I deliberately do not include version numbers or specific dates), and readers should bear in mind that models in 2025, let alone 2030, may produce different outputs. However, the high-level points I am making should still be valid.

I also focus in this book on my own experiences. Where possible I use examples from systems which I have worked on or otherwise been involved with, even if they are not the best known system in their area; for instance for this reason I talk

about the (somewhat obscure) BLOOM language model as well as better-known ones such as GPT. More generally the book focuses on data-to-text NLG (systems which use NLG to summarise and explain non-linguistic data) because this is my personal interest. I also include personal notes throughout the book. I hope this personal focus makes the book more interesting to readers.

Specifically, the book has the following **chapters**:

- *Introduction*: I present some example NLG systems, summarise the content of the rest of the book, and also give a short history of NLG.
- *Rule-based NLG*: I describe how AI systems can generate texts using algorithms and rules which explicitly make decisions about the content and language of generated texts. Rule-based NLG has been overshadowed by neural NLG in recent years, but it is still the best way to build some NLG applications. Rule-based NLG also shows the types of decisions which need to be made in text generation, and I think a good understanding of this helps anyone working in NLG, even if they use other approaches.
- *Machine learning and neural NLG*: I give an overview of machine learning and neural approaches to NLG, including language models. This area is changing very rapidly, and models which were exciting state-of-the-art a few years ago are now obsolete and forgotten. Because of this, I just give a high-level overview of basic concepts behind models, and then discuss data and other issues which are important regardless of the model used.
- *Requirements*: As with any type of software, knowing what users and stakeholders are looking for is essential in building a successful NLG application. I look at some of the different quality criteria that people may care about, workflows for using NLG (including ‘human-in-loop’), textual vs graphical presentation of information, and methodologies for understanding (acquiring) requirements.
- *Evaluation*: This is the longest chapter of this book, which reflects my interest in the topic as well as its importance. From both a scientific and practical perspective, it is essential to evaluate how well NLG algorithms, models, and techniques work, using experiments which are rigorous and replicable. I discuss basic evaluation concepts, and then describe techniques for human and automatic (metric) evaluation. I also look at evaluating real-world impact of NLG systems, as well as commercial evaluation.
- *Safety, testing, and maintenance*: Society expects that AI systems used in the real-world will be safe (not harm users or third parties); systems which are not safe will not be allowed by governments and regulators. I examine safety concerns and techniques in NLG, and also look at software testing, which is used to identify bugs and other problems which could lead to unacceptable behaviour. I conclude with a section on maintaining NLG systems, which is very important (most of the lifecycle costs of software systems are in maintenance) but poorly understood.
- *Applications*: NLG is not just an academic discipline, it is also a technology which can be used to build useful applications which help people. In this chapter I discuss some fundamental issues (such as scalability), and then look in more

detail at four areas which NLG has been used commercially for many years: journalism, business intelligence, summarisation, and medicine. Lessons from these long-standing NLG use cases can be applied to newer applications of NLG.

It is impossible for me to **acknowledge** all of the people who have helped me in my NLG career. I would like to give special thanks to my faculty colleagues in the Aberdeen NLG group over the years, including Kees van Deemter, Jim Hunter, Ruizhe Li, Chengua Lin, Judith Masthoff, Chris Mellish, Graeme Ritchie, Advait Siddharthan, Arabella Sinclair, Yaji Sripada, and Wei Zhou. I would also like to thank the many PhD students and post-docs I have worked with at Aberdeen, including Jawwad Baig, Simone Balloccu, Daniel Braun, Martin Dempster, Albert Gatt, Rodrigo Gomes de Oliviero, Francois Portet, Stephanie Inglis, Dave Howcroft, Kitt Kuptavanich, Jing Lin, Saad Mahamood, Meg Mitchell, Wendy Moncur, Francesco Moramarco, Giulia Pucci, Joe Reddington, Roma Robertson, Jaime Sevilla, Adarsa Sivaprasad, Mengxun Sun, Barkavi Sundararajun, Iniakpokeikiye Thompson, Craig Thomson, Nava Tintarev, Ross Turner, Sandra Williams, and Jin Yu. The Aberdeen University NLG group is by no means the largest NLG group in the world, but it has provided a great environment for me to pursue my interests.

Of course funding is very useful for university research, and I am very grateful to Australian Research Council, Cancer Research UK, the EU Horizon and Marie-Curie programmes, the Royal Society of Edinburgh, the Scottish Chief Scientist Office, and the UK Economic and Social Research Council for their support. Special thanks to the UK Engineering and Physical Sciences Research Council, which has funded my projects over a 30-year period. I am also grateful to IBM and the US National Science Foundation for supporting my PhD studies at Harvard.

In 2009 I co-founded a company, Data2Text, which was acquired by Arria NLG in 2013. I worked at Data2text and Arria for many years, and my experiences there shaped much of the content of this book. At Arria, I am especially grateful to John Alexander, Neil Burnett, William Bradshaw, Robert Dale, Sharon Daniels, Ian Davy, Jay Dewalt, Cathy Herbert, Alasdair Logan, Lyndsee Manna, Daniel Paiva, John Perry, Kapila Ponnampereuma, Jette Viethen, and Keith Wisemann for their help in understanding the commercial perspective on NLG, as well as many colleagues from Aberdeen University (mentioned above) who spent time at Data2text or Arria.

Numerous people have helped me with this book. I would especially like to thank the people who reviewed and gave comments on draft material: Kees van Deemter, Nick Diakopoulos, Albert Gatt, Emiel Krahmer, Emiel van Miltenburg, Verena Reiser, and Ross Turner. Extra special thanks to Saad Mahamood, who reviewed and commented on a draft of the complete book. Of course, I alone am responsible for any mistakes in this book!

Finally, I am grateful to my wife Ann and my children Miriam, Moshe, and Naomi for putting up with me while I worked on this book!

Contents

- 1 Introduction to NLG** 1
 - 1.1 What Is Natural Language Generation?..... 2
 - 1.2 Example: Weather Forecast (*Data-to-Text*) 3
 - 1.2.1 Use Case: Point Weather Forecasts for General Public 4
 - 1.2.2 Technology: Rule-Based NLG 5
 - 1.2.3 Evaluation 6
 - 1.3 Example: Summarising Consultations (*Text-to-Text*)..... 6
 - 1.3.1 Use Case: Generating Summaries of Doctor-Patient Consultations..... 6
 - 1.3.2 Technology 8
 - 1.3.3 Evaluation 8
 - 1.4 Technologies 9
 - 1.4.1 Rule-Based NLG..... 10
 - 1.4.2 Machine Learning and Neural Models..... 11
 - 1.4.3 Combining Rules and ML 12
 - 1.5 Effectiveness 13
 - 1.5.1 Requirements 13
 - 1.5.2 Evaluation 14
 - 1.5.3 Safety, Testing, and Maintenance 15
 - 1.6 Use Cases and Applications 16
 - 1.7 Ethics 18
 - 1.8 A Very Short History of NLG 19
 - 1.8.1 Early History 19
 - 1.8.2 1990–2014 20
 - 1.8.3 2015–2024 20
 - 1.8.4 My Personal NLG Journey 22
 - 1.9 Resources and Further Reading 22
- 2 Rule-Based NLG** 25
 - 2.1 NLG Pipeline..... 26
 - 2.2 Examples 27

2.2.1	DrivingFeedback	28
2.2.2	Babytalk	29
2.3	Signal Analysis	30
2.3.1	Noise Detection: Principles	31
2.3.2	Pattern Detection: Principles	31
2.3.3	Techniques for Signal Analysis	33
2.4	Data Interpretation	33
2.4.1	Principles	33
2.4.2	Techniques	34
2.5	Document Planning	35
2.5.1	Principles	35
2.5.2	Techniques	36
2.6	Microplanning	37
2.6.1	Lexical Choice	37
2.6.2	Generating Referring Expressions	39
2.6.3	Aggregation	40
2.7	Surface Realisation	41
2.7.1	Principles	41
2.7.2	Techniques	42
2.8	Template NLG	43
2.8.1	Principles	44
2.8.2	Techniques	45
2.9	Further Reading and Resources	46
3	Machine Learning and Neural NLG	49
3.1	Examples	50
3.1.1	Very Simple Trained Model: a vs. <i>an</i>	50
3.1.2	Fine-Tuned Neural Model: Facebook Weather Dialogues	52
3.1.3	Prompted Model: Using ChatGPT to Generate Weather Forecasts	53
3.2	Machine Learning Models for NLG	55
3.2.1	Classifiers	55
3.2.2	N-Gram Language Models	56
3.2.3	Early Neural Models	57
3.2.4	Transformers and Foundation Models	57
3.2.5	Instruction Tuning and RLHF	58
3.2.6	End-to-end vs. Modular Architectures	60
3.3	Training Data	60
3.3.1	Data Sources	61
3.3.2	Data Set Criteria	62
3.3.3	Impact of Training Data on Prompted Models	63
3.3.4	Synthetic Data and Data Augmentation	64
3.3.5	Including Examples in Prompts	65
3.4	Issues	65
3.4.1	Domain Shift	66

3.4.2	Question Answering	67
3.4.3	Auditability and Controllability	68
3.4.4	Legal and Regulatory Issues	68
3.5	Further Reading and Resources	69
4	Requirements	71
4.1	Quality Criteria: Texts	72
4.1.1	Readability and Fluency	73
4.1.2	Accuracy	74
4.1.3	Content	76
4.1.4	Utility	76
4.2	Quality Criteria: Systems	77
4.2.1	Non-functional Requirements	77
4.2.2	Consistency and Variation	78
4.2.3	Average vs. Worst Case	79
4.3	Workflow	80
4.3.1	Fully Automatic NLG	80
4.3.2	Human Checking and Editing	80
4.3.3	Creating Drafts for Human Writers	81
4.4	Text and Graphics	82
4.4.1	Decision Support	83
4.4.2	Other Use Cases	84
4.4.3	Combining Text and Graphics	84
4.5	Requirements Acquisition	85
4.5.1	User Studies	85
4.5.2	Manual Corpus Analysis	88
4.5.3	Stakeholders	91
4.6	Further Reading	92
5	Evaluation	93
5.1	Example: Smoking Cessation (Impact Evaluation)	94
5.2	Fundamentals	96
5.2.1	Stakeholder Perspective	96
5.2.2	Hypothesis Testing	97
5.2.3	Statistical Hypothesis Testing	98
5.2.4	Experimental Design, Execution, Reporting, and Follow-Up	99
5.2.5	Research Questions	101
5.2.6	Replication	101
5.2.7	Ecological Validity: Artificial Versus Real-World Context ...	103
5.2.8	Test Data: Representative, Different from Training Data	104
5.3	Human Evaluation	105
5.3.1	Types of Human Evaluation	105
5.3.2	Experimental Design	110
5.3.3	Issues in Human Evaluation	118
5.4	Automatic Evaluation	121

5.4.1	Types of Automatic Evaluation	122
5.4.2	Experimental Design	124
5.4.3	Examples of Metrics	128
5.4.4	Validation of Metrics	130
5.5	Impact Evaluation	134
5.5.1	Comparison Between Users: Randomised Controlled Trials and A/B Testing	134
5.5.2	Historical Comparison	134
5.5.3	Challenges	135
5.6	Commercial Evaluation	136
5.6.1	Costs	136
5.6.2	Benefits	137
5.6.3	Risks	137
5.6.4	Return on Investment (ROI)	138
5.7	Ten Tips on Evaluating NLG	139
5.8	Further Reading	140
6	Safety, Testing, and Maintenance	143
6.1	Safety	143
6.1.1	Safety Concerns in NLG	144
6.1.2	Approaches to Addressing Safety Concerns	148
6.2	Software Testing of NLG Systems	151
6.2.1	Testing Systems with Variable Outputs	152
6.3	Maintenance	152
6.3.1	Changes in Domain and User Needs	154
6.3.2	New Users and Use Cases	155
6.3.3	Changes in Models	156
6.4	Further Reading and Resources	157
7	Applications	159
7.1	Key Attributes of Successful NLG Applications	159
7.1.1	Volume and Scalability	159
7.1.2	Data Availability	160
7.1.3	Accuracy	161
7.1.4	Maintainability and Adaptability	161
7.1.5	Acceptability and Trust	163
7.1.6	Conforming to Genre and Sublanguage	164
7.2	Journalism	164
7.2.1	Example: BBC Election Reporter	165
7.2.2	Types of News	166
7.2.3	Fake News	168
7.3	Business Intelligence	169
7.3.1	Example: Covid Reporter	169
7.4	Summarisation	170
7.4.1	Example: Summarising Emails with Google Bard	171
7.5	Medical Applications	172

7.5.1	Use Case: Reporting	173
7.5.2	Use Case: Patient Information and Behaviour Change	175
7.5.3	Use Case: Clinical Decision Support	176
7.5.4	Medical Business Intelligence Use Cases	177
7.5.5	Safety	178
7.6	Further Reading	179
References		181
Index		197

Chapter 1

Introduction to NLG



This book gives my perspective on Natural Language Generation (NLG), that is using artificial intelligence (AI) techniques to produce texts in English and other human languages; it draws on my decades of experience in the field. I wrote a book (with Robert Dale) on NLG in 2000 [162], when I had limited experience building NLG systems. Over the past 25 years I have gained a lot more experience building systems, and this book largely reflects the issues and challenges I faced; most of the examples come from systems I have worked on. It draws on my experiences at the company Arria NLG (which evolved from a spinout company, Data2text, which I co-founded in 2009) and my academic research at the University of Aberdeen.

This book is intended to help students, developers, and researchers understand the fundamental concepts of NLG. It is not intended to be an up-to-date guide to the latest language model technologies, research papers, and commercial offerings. In particular, the book discusses technology only at a high level because technology changes very fast; hence detailed material on technology would quickly become out-of-date. The book's main focus is on fundamental aspects of NLG which change more slowly than technology, including component tasks of NLG, requirements of NLG systems, evaluation, safety and testing, and applications. These topics are essential for success (at least in my experience) but are less often discussed than technology; I hope this book can help readers understand these aspects of NLG.

The book focuses on systems which generate output texts that describe, summarise, explain, etc. the system's input data. In other words, from the perspective of generative AI systems such as ChatGPT and Google Gemini, it focuses on applications where an input is transformed into an output (see examples later in this chapter), not applications where the user asks a question and the system synthesises an answer from Internet sources

1.1 What Is Natural Language Generation?

Natural Language Generation systems use artificial intelligence and natural language processing techniques within software systems that generate texts in human languages such as English, Chinese, and Arabic. In other words, NLG is the science of AI systems that can write. As such it is related to (but not the same as) Natural Language Understanding (NLU), which is the science of AI systems that can read and extract meanings from human-written texts.

Natural Language Generation has recently become more prominent because of the success of ChatGPT and other generative language models, but the field has been around for decades. AI and NLP techniques such as machine learning and language models are widely used, but they do not tell the whole story. NLG researchers and developers should understand (Fig. 1.1):

- *AI and NLP techniques*: technology used to generate texts.
- *Human-computer interaction (HCI)*: how NLG systems can effectively interact with users.
- *Linguistics and psychology*: how to best use language to communicate with users.
- *Software engineering*: how to build real-world usable software systems.
- *Domain knowledge*: what is important in the target domain, and how it should be communicated.

Personal Note

I have found that research is much more rewarding if it takes the larger context into consideration. Certainly in my own career I have enjoyed learning about

(continued)

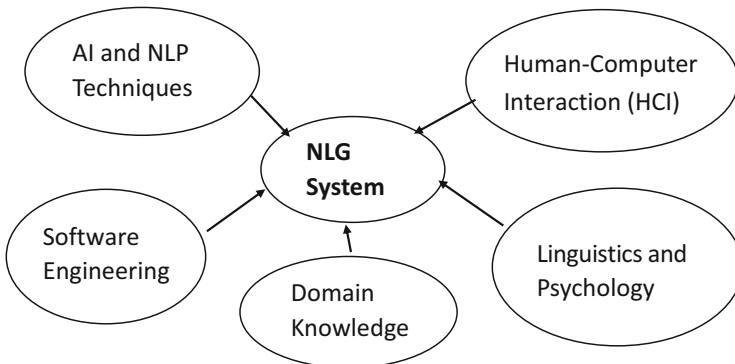


Fig. 1.1 Skills needed for NLG

the above areas and collaborating with all sorts of people, ranging from HCI and usability experts to domain experts in offshore oil platforms. Many of my most influential research papers have benefitted from collaborations with and insights from HCI experts [95], psychologists and linguists [129], software engineers [159], and domain experts [148].

From a practical perspective, NLG is used in many different contexts. My focus in this book is on systems that generate texts which describe, explain, summarise (etc.) an input to the system. The input can be numbers, in which case the NLG system is a *data-to-text* system (see example in Sect. 1.2); the input can also be words, in which case the NLG system is a *text-to-text* system (see example in Sect. 1.3). Machine translation (MT) is sometimes considered to be a type of NLG, and I will occasionally refer to MT, especially when discussing evaluation. However, readers interested in MT should look at a specialised book such as Koehn’s *Neural Machine Translation* [100]. I also will not discuss writing assistants (sentence completion, grammar improvements, etc.) or tools for generating fictional content. I discuss using generative language models to produce text, but I do not discuss other capabilities of these models such as language understanding and image generation.

1.2 Example: Weather Forecast (*Data-to-Text*)

One of the earliest use cases for NLG was generating textual weather forecasts from numerical weather predictions. The numerical weather predictions are produced by atmosphere simulations and give predicted weather parameters (e.g. wind speed, precipitation) at different points and times. The NLG system takes this information and produces a textual forecast which summarises in words how the weather will develop. As described below, using NLG to produce forecasts makes it easier to produce specialised forecasts from the same numerical weather prediction data, for example forecasts for different locations or for different types of users.

Many NLG weather forecast systems have been developed over the years, with the first production system entering operational usage in 1992 [71] (Fig. 1.7). More recently there has been considerable interest in using neural NLG techniques to allow dialogue systems to respond to weather queries (Sect. 3.1.2).

Forecast generation is a type of *data-to-text* NLG, that is a system that generates text that summarises a numeric (or symbolic) data set. In this case, the input data is a large set of numbers that describe wind speed, precipitation, etc.; symbols are used to describe some weather parameters such as wind direction (E, NE, N, NW, etc.). For some types of weather forecasts, the input data set can consist of millions of numbers and symbols [201]. The NLG system ‘digests’ and analyses this data and generates a text that summarises key facts and insights for the user.

Because NLG researchers have worked on generating weather forecasts for over 30 years, it is a good domain to understand NLG issues and technologies, and this book will refer to weather forecast generation in many places.

1.2.1 Use Case: Point Weather Forecasts for General Public

My company, Arria, developed a simple forecast generator for the UK Met Office [186] in 2014. It is by no means the most sophisticated NLG system in this space, but it does illustrate weather forecast generation and data-to-text NLG. The Arria system is not being used at the time of writing, but rule-based techniques are still widely used to produce simple weather forecasts.

This system generates a *point forecast*, that is a weather forecast for a particular location over a time period (such as Heathrow airport in London). *Area forecasts*, in contrast, describe the weather over a region (such as Scotland); NLG systems that generate area forecasts (such as [201]) tend to be considerably more complex than systems which generate point forecasts.

The Arria system produced *public forecasts*, that is forecasts that are intended for the general public. There are also of course many specialised forecasts, for farmers, pilots, sailors, etc.; these emphasise different types of information (e.g. aviation forecasts describe wind speeds at different altitudes; public forecasts do not). They also often used specialised terminology (e.g. marine forecasts use the verb *backing* to describe a counter-clockwise change in wind direction).

The commercial goal of the Arria system was to produce weather forecasts for 5,000 different locations; this would be a daunting task for human forecasters. In general the benefits of NLG are often in allowing large numbers of specialised texts to be produced, so that users can receive texts that are specific to their needs, instead of generic texts. In other words, an Aberdeen-specific forecast is of more use to me than UK-wide forecast, but producing textual forecasts for the thousands of towns and villages in the UK is difficult without automation.

Figure 1.2 shows an example public forecast produced by the Arria system for London's Heathrow Airport, and some of the data used to produce this forecast. The forecast itself is

Sunshine from mid-morning and into the afternoon. Staying dry, but becoming cloudier from early evening and into Thursday. It is likely to feel milder than on Tuesday with a maximum temperature during the afternoon in the region of 11C and a minimum temperature overnight of around 6C. Light winds throughout.

The data used to produce the forecast (i.e. the output of the numerical weather model) is a table that shows the values of different weather parameters at different times, for this location. Part of the table is shown beneath the forecast in Fig. 1.2, for example the temperature at 0000 is 4°C.

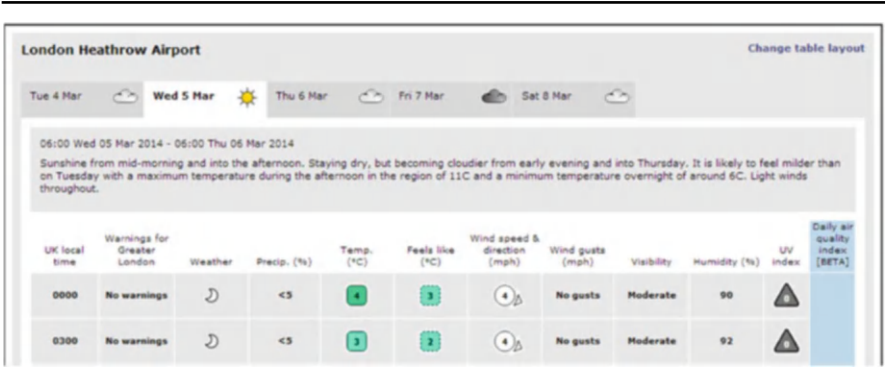


Fig. 1.2 Weather forecast from Arria system; Figure 1 from [186]

1.2.2 Technology: Rule-Based NLG

The Arria system uses rules to generate the text (not machine learning). Details are not given in [186], but in rough terms the rules do the following:

- *Stage 1: Find patterns in data (signal analysis).* For example the probability of precipitation is very low (less than 5%) for the entire period.
- *Stage 2: Generate insights (data interpretation).* For example this will be a dry day. Insights depend on what the user is interested in. For instance, as mentioned above, an aviation forecast will describe wind speed at different altitudes, since this is very important to pilots. In contrast, a public forecast probably will only describe wind speed at ground level, since this is all the general public cares about.
- *Stage 3: Decide which insights to present, how they are ordered, and how they are related to each other (document planning),* for example adding a contrast link (expressed as *but*) between the low-precipitation insight and the insight that cloud cover increases.
- *Stage 4: Decide how to express insights linguistically, in a stylistically appropriate manner (microplanning),* for example deciding to use the gerund phrase *staying dry* to communicate the low-precipitation insight.
- *Stage 5: Generate grammatically correct text (surface realisation),* for example capitalising *Staying* since it is the first word of the sentence.

All data-to-text systems must perform the above types of processing in some fashion. That is they must find patterns in the data, generate insights that are useful to the user, order and link insights, express insights as words, and deal with grammatical details. The Arria system does these steps explicitly, using rules.

At the other extreme, a neural model can be trained to go from input data to output text in a single step, but this model would still need to make decisions

about patterns, insights, document structure, linguistic expression, and grammar conformance, even if these decisions are buried in a neural network instead of expressed explicitly as rules. Another approach is to mix technologies, for example to use algorithms and rules to find patterns and generate insights and a neural model to select and express insights in grammatically correct text.

1.2.3 Evaluation

The Arria system was evaluated by putting it onto a website used by the Met Office to test new technologies and asking users to rate the usefulness and desirability of the forecasts; this is an example of a *human evaluation based on ratings* (Sect. 5.3.1.1). Users who submitted ratings were overall very positive; however we need to keep in mind that people who looked at the Met Office ‘new technology’ site may not be representative of Met Office users more generally. User ratings are often used to evaluate NLG systems, but it is considered best practice to try to get a diverse and representative set of users to do this.

This and other points related to evaluation of NLG systems are discussed in detail in Chap. 5.

1.3 Example: Summarising Consultations (*Text-to-Text*)

Weather forecasts are an example of data-to-text NLG, where the input to the NLG system is structured data. In contrast, *text-to-text* NLG systems generate texts from other texts. One example is *summarisation* systems, which generate texts that summarise input texts.

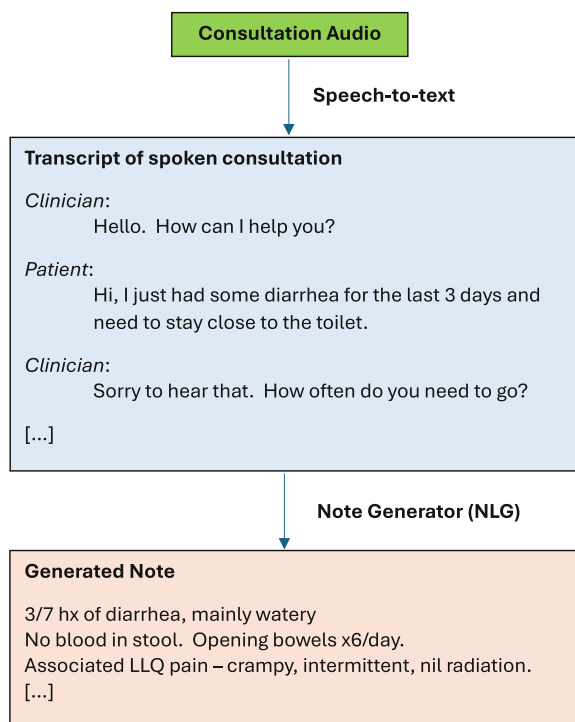
A concrete example is generating texts that summarise consultations between professionals and their clients. Such summaries are often useful as a record of what was discussed and decided; in some contexts they are legally required.

1.3.1 Use Case: Generating Summaries of Doctor-Patient Consultations

Babylon, a UK digital health company, developed the Note Generator system [95] which generated summaries of doctor-patient consultations. In the UK and many other countries, whenever a doctor talks to a patient, a summary of this consultation needs to be added to the patient’s medical record. In the past these summaries were written manually, either by the doctor or by a medical scribe who listens to a recording of the consultation. Interest is growing in automating at least part of the process of summarising a consultation.

Two goals of the Babylon Note Generator system are to (A) reduce the amount of manual effort required to create the summary and (B) enable doctors to focus on the patient on the consultation without the distraction of taking notes [95]. Note

Fig. 1.3 Example of Note Generator; based on Figure 1 from [95]



Generator is no longer used,¹ but similar systems are being developed and used by other healthcare companies at the time of writing, such as Abridge.²

A simple example is shown in Fig. 1.3. The consultation is recorded and transcribed by a speech-to-text system; a short extract of a transcript is shown in the Transcript box in Fig. 1.3. The NLG system then produces a summary of the transcript for the patient record (shown in the Generated Note box in Fig. 1.3); this communicates key medical information in standardised medical language. The summary is shown to the doctor, who can edit it to add missing information, correct mistakes, etc. before the summary is added to the patient record.

Of course some consultations are more difficult than others; for example it is difficult to write up a consultation when the doctor suspects that the patient may not be telling the truth. Note Generator focuses on summarising straightforward consultations (which are the majority); doctors still need to manually write notes for difficult cases, especially if information needs to be added beyond what was said in the consultation. This is common in NLG and indeed AI; the NLG system helps

¹ Babylon unfortunately went bankrupt in 2023.

² <https://www.abridge.com/>

the user with ‘straightforward’ cases but provides less assistance in more complex, unusual, or novel situations.

1.3.2 Technology

The Note Generator system uses *neural language models* to generate the summaries. A neural language model essentially is a machine learning system which is trained on (input, output) pairs to perform a task. For the Note Generator, the developers started with a standard model used for summarisation (BART [113]) which had been trained on almost 300,000 pairs of news articles and their summaries. Of course summaries of medical consultations are very different from summaries of news articles, so the developers adapted the model by *fine-tuning* it on 10,000 examples of previous doctor-patient consultations which had already been manually summarised and entered into Babylon’s patient record system. This is a common strategy in neural NLG; we train a model on a large data set which is only loosely related to the application and then use a smaller data set to adapt the model to the target domain and application.

From a summarisation perspective, one unusual aspect of Note Generator was that doctors wanted summaries to be generated incrementally, so that at any point in time there was a summary of the consultation so far [95]; this is an example of a human-computer interaction constraint (Fig. 1.1). Previous summarisation systems generated a complete summary from a complete input document, so the team essentially trained the model to generate incremental additions to an existing summary. The model always added content to the summary, it never changed what the doctors had already seen. This was novel technology (the team applied for a patent) and shows how taking HCI considerations seriously can lead to new NLG technology.

1.3.3 Evaluation

The Note Generator System was evaluated in a number of different ways [95, 137, 138]:

- *Automatic metrics*: The system was run on a *test set* of historical consultations (which had not been used to train or fine-tune the model), and several different metrics (algorithms) were used to assess how similar the system’s output was to the manually written summary in the historical report. This is a very common type of evaluation in NLP, which is discussed in Sect. 5.4. Such evaluations are easy to do but unfortunately are not always meaningful. In this case, the scores returned by the metrics were not good predictors of actual utility [138].
- *Error analysis*: The team asked doctors to find mistakes in the generated summaries, such as incorrect statements, omitted content, and misleading statements.

- This information was very useful in itself for understanding where the system worked well and where it did not, and it was also a good predictor of utility. Evaluations via error analysis are discussed in Sect. 5.3.1.2.
- *Post-edit time*: The team asked doctors to edit summaries and fix mistakes and timed how long this took; post-editing time is one of the most important KPI (Key Performance Indicator) for real-world success. Not surprisingly, the time taken to post-edit was correlated with the number of errors. This is an example of a *task-based* evaluation, where we measure the impact an NLG system has on human task performance; such evaluations are discussed in Sect. 5.3.1.3.
 - *Real-world utility*: Last but certainly not least, after the system was deployed and used in real consultations, the team measured how long it took doctors to post-edit notes in real consultations and how this compared to the time needed to write notes manually before the system was deployed. They also assessed the number of errors in post-edited Note Generator summaries (from real consultations) and compared this to the number of errors in manually written summaries. Evaluations of real-world utility and impact are discussed in Sect. 5.5.

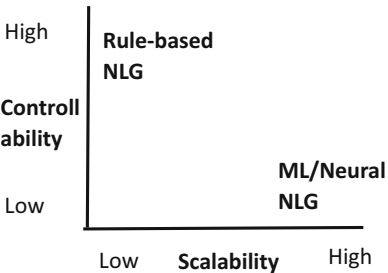
Using a broad mix of evaluation techniques is good practice and gives us a better perspective on how well a system is performing.

1.4 Technologies

The first part of this book discusses technologies used to build NLG systems. Because NLG technology is changing very fast, the discussion is high level and conceptual; readers interested in details of the latest research paper or commercial product should look at other sources for this information.

At the most basic level, there are two ways of building NLG functionality: We can write *rules* and code which execute NLG algorithms, or we can use *machine learning* models (usually based on neural deep learning technology), which may be trained or fine-tuned on the application domain. In very crude terms, these technologies involve different trade-offs between the following (Fig. 1.4):

Fig. 1.4 Scalability and controllability of different NLG technologies



- *Scalability*: How much effort is required to build a large-scale NLG system which produces sensible output texts for a wide variety of inputs?
- *Controllability*: Can developers and users precisely control what the NLG system does? A related criterion is *Transparency*; do developers and users understand how and why the NLG system produces a text?

Rule-based NLG systems are very controllable and transparent, but large-scale wide-coverage NLG systems may require thousands of rules; creating these rules is expensive and time-consuming. Neural NLG systems, in contrast, are black boxes which are difficult to control and not transparent, but building such systems can be relatively cheap and quick if appropriate training data is available.

Of course this is a very over-simplified comparison. For example software engineering tells us that most of the life-cycle costs of a successful software product are in maintenance, not initial development, and we do not have a good understanding of how the cost of maintaining a rule-based system is compared to the cost of maintaining a neural system. Maintenance of NLG systems is discussed in Sect. 5.5.

1.4.1 Rule-Based NLG

Chapter 2 looks at rule-based NLG systems, which execute specific algorithms to analyse data, extract insights, choose the content of generated texts, choose words and syntactic structures, and generate linguistically correct output texts (an example was given in Sect. 1.2.2). The chapter focuses on data-to-text because rule-based approaches are rarely used in text-to-text applications. Rule-based systems often use a modularised *pipeline* architecture (Sect. 2.1), where different modules are used for each of these tasks (e.g. pattern detection is done by a module specialising in this task), and these modules are usually run sequentially (i.e. all pattern detection is done, then all insight generation is done, etc.). Often there is a core set of libraries and modules which encodes key analytics and linguistic processing algorithms, which is supplemented with a layer of rules that encode domain knowledge for a specific application.

From a high-level perspective, rule-based systems have the advantage of being precise and modifiable; if the system is not doing what the user wants, developers can modify the rules to match user needs. They are also transparent and auditable and can be tested using software engineering techniques; this is important in many commercial contexts, especially in contexts where companies are legally liable if an NLG system makes a mistake. Finally, rule-based systems are good at integrating analytics (insight generation) and linguistic (insight expression) reasoning and in incorporating domain/genre knowledge into both types of reasoning. Their main disadvantage is that writing rules is time-consuming (and can require specialised domain knowledge), especially for systems that have broad or changing use cases.

For these reasons, rule-based systems work well for business-oriented data-to-text NLG systems in focused domains and use cases where the output needs to be accurate.

1.4.2 Machine Learning and Neural Models

Chapter 3 discusses how *machine learning*, especially *neural deep learning* technology, can be used to create models that produce texts. This often requires collecting a training set (sometimes called *corpus*) of example input-output pairs and using this to create a model for the task; the model essentially learns from the corpus how to map system inputs into textual outputs.

Training a high-quality language model from scratch requires a lot of training data, typically at least a billion tokens (words). Most NLG applications do not have this much training data available; for example only 10,000 input-output pairs (consultation transcript and corresponding human-written summaries) were available to train Note Generator (Sect. 1.3). Fortunately, we can also build good models by *fine-tuning* general-purpose *large language models (LLMs)* (these are sometimes called *foundation* models). In other words, we can take a large generic model such as BART [113] or T5 [154] which has been trained on billions of tokens of generic Internet material and then adapt the model to a specific task using a smaller corpus of actual input-output pairs for this task. This was the approach used in Note Generator; the 10,000 pairs of transcripts and corresponding human-written summaries were not sufficient to train a good model from scratch, but they were sufficient to fine-tune a large generic model (BART).

Some LLMs, such as ChatGPT, can be used without training or fine-tuning corpora, by simply asking the model (via a *prompt*) to generate a text. For example in order to summarise a text, we could give ChatGPT a prompt that says something like *Summarise the following text*, followed by the text to be summarised (Fig 1.5). In this example the summary is not very good (almost as long as the original text), but it was generated without needing any application-specific or domain-specific training or fine-tuning. We can optionally include a small number of examples (5 or 10, not 10,000) in the prompt in order to give the model some guidance as to what we are looking for; we can also use *prompt engineering* to tune the wording of the prompt in order to get a more appropriate result.

From a high-level perspective, the great advantage of ML approaches is that they can be used to build wide-coverage NLG systems, without needing to write a large number of rules. Neural NLG systems are also very good at generating readable and fluent texts. A substantial amount of training data is needed to fine-tune models (and much more data to train models from scratch), but we do not need training data if we can simply prompt an existing model to do the task (as in Fig. 1.5).

On the other hand, neural NLG systems are difficult to control; for example if the user wants the output texts to be modified in specific ways, this is not straightforward

Prompt: (input to ChatGPT)

Summarise the following text:

I think journals do a much better job than conferences at reviewing papers. This year I got a good perspective on this issue, because I was both an Action editor at TACL (Transactions of the ACL), which is a leading NLP journal; and a Senior Area Chair for the ACL conference, which is a leading NLP conference. The experience reinforced my belief that reviewing is better at journals, and I summarise some of my observations below.

Result: (from ChatGPT)

The author believes that journals are more effective than conferences when it comes to reviewing papers. Their perspective on this matter was strengthened by their dual role as an Action editor at TACL, a reputable NLP journal, and a Senior Area Chair for the ACL conference, a prominent NLP conference. Based on their experience, they summarize their observations supporting the superiority of journal reviews.

Fig. 1.5 Using a large language model (ChatGPT) for NLG

with neural models. Sometimes this can be done by modifying a prompt, but this is not always sufficient. Another major problem is that neural models can generate texts which are factually inaccurate or otherwise have inappropriate content. A related point is that software testing and quality assurance (Sect. 6.2) of neural systems is challenging, since they are not transparent and understandable. For this reason, neural NLG is often used in *human-in-the-loop* workflows (Sect. 4.3), where a person checks and fixes the model's output (as is the case with Note Generator), at least in contexts where accuracy is very important. However human-in-the-loop may not be necessary in contexts where accuracy is not of paramount importance, such as generating fiction or advertising material.

1.4.3 Combining Rules and ML

It is of course possible to build NLG systems that combine rule-based processing and machine learning. One approach is to use a pipeline architecture but have some of the processing within modules done by neural systems [33]. For example we can use rules where precision and accuracy are very important (e.g. selecting key insights to communicate) and neural techniques where fluency is very important (e.g. choosing syntactic structures of sentences).

A variant of this is to generate an initial draft text using rules, which is guaranteed to have the right content (insights) but may not be very readable, and use a language model to rewrite the text to make it more fluent and readable [89].

Another approach is to build two systems, one neural and one rule-based, run both, and use a selection module to decide which of the two outputs to use. For example the selection module could test the text produced by the neural system for

accuracy errors and use the neural text if none were detected and the rule-based text otherwise [78].

1.5 Effectiveness

The second part of this book looks at the usefulness and effectiveness of NLG systems. It starts by looking at *requirements*, in other words what NLG systems need to do in order to be useful. It then looks at *evaluation*, i.e. measuring how well NLG systems meet important quality criteria (as determined by requirements). It concludes by looking at *safety and testing*, which focus on whether the system behaves unacceptably in a few cases even if it is fine in most cases.

1.5.1 Requirements

Chapter 4 looks at requirements for NLG systems; what do real-world users want NLG systems to do? Where does NLG ‘add value’ in real-world use cases, and how does it impact workflows? What specific NLG capabilities are needed by users?

Questions like the above are often ignored by academic researchers, but they are very important for real-world success. Just as with other types of software, an NLG system is not going to succeed if we do not understand user requirements and more generally how users will react to the NLG system. Some of the key requirements issues discussed in Chap. 4 are:

- *Quality Criteria*: What are the text (and system) characteristics which may be important to users and other stakeholders and which of these matter most in different contexts? For example when is accuracy more important than readability? Also, are users only interested in average text quality or do they need a guarantee that all texts will meet a minimum quality threshold?
- *Workflows*: While some NLG systems operate without human supervision, many NLG systems collaborate with humans and as such are integrated with human workflows. What are different ‘human-in-the-loop’ workflows and when do they make sense?
- *Text and Graphics*: Computer systems can communicate information and otherwise interact with users using visualisations and interactive data graphics as well as text. When is text the best interaction and communication modality? How should NLG texts should be combined and integrated with visualisations?
- *Understanding Requirements*: How do NLG developers work with users and other stakeholders in order to understand their needs? Many techniques for requirements acquisition have been developed in other parts of computing science; how can we adapt these to work for NLG [95]?

1.5.2 Evaluation

Chapter 5 discusses evaluation of NLG systems. From a scientific perspective, we need to be able to evaluate NLG models, algorithms, and systems, in order to understand how they compare to previous work and also where further work is needed.

Scientific evaluation is essentially hypothesis testing. In very general terms, we can evaluate NLG systems by asking people to examine or use generated texts (*human evaluation*), by using *automatic metrics* (algorithms) to assess the quality of texts, or by assessing the system's real-world impact when deployed (*impact evaluation*). For all types of evaluation, a key question is how meaningful it is, and in particular whether the evaluation is a good predictor of real-world utility.

There are many kinds of human evaluation. The most expensive, and usually the most meaningful, are *task-based evaluations*. Such evaluations involve asking domain experts to perform a task using the generated text, such as asking doctors to decide on a medical intervention for a patient after reading an NLG summary of the patient's status [148]. A more common approach (which is not coincidentally considerably cheaper and quicker) is to ask non-expert subjects to read generated texts and either rate them on quality factors such as accuracy and readability or rank (order) texts based on a quality factor. Such studies need to be carefully designed in order to be meaningful [203]. Another approach to human evaluation is to ask subjects to annotate errors and other problems in generated texts; this is often more meaningful than rating-based evaluation but can also be more expensive.

Metric (algorithmic) evaluations are usually much cheaper and faster than human evaluation; because of this, metric evaluations can be done more frequently and on a larger scale than human evaluation (a typical academic human evaluation may look at 100 texts; a metric evaluation can look at many thousands or even millions of texts). At the time of writing, most metric evaluation is *reference-based*, that is generated texts are algorithmically compared against human-written 'gold standard' *reference* texts. Numerous such metrics have been designed, and they are usually *validated* by seeing how well they correlate with high-quality human evaluations. Correlation may depend on factors such as text quality; for example a metric may correlate well with human evaluations of low-quality NLG texts but not of high-quality NLG texts. There are also *referenceless* metrics that assess the linguistic quality of a text in isolation, including well-established formulas such as Flesch-Kincaid scores. Interest is growing in using large language models such as GPT to evaluate texts (both with and without references); this seems to work well in many contexts.

Impact evaluation is perhaps the hardest to carry out, not least because evaluating a system in production usage may raise ethical issues; for example if we evaluate a medical decision support system in real clinical usage, we need to show beforehand that the system will not damage patient care. The most rigorous impact evaluation is to conduct a *comparison experiment* where some people use the NLG system and others use an alternative system and compare outcomes; this is related to *A/B*

Testing. An alternative is a *historical comparison* where an NLG system is deployed to a set of users, and *Key Performance Indicators (KPIs)* are compared before and after the system is introduced.

1.5.3 Safety, Testing, and Maintenance

Developers of real-world NLG systems need to ensure that their systems are safe (never harm users), properly tested from a software perspective (including edge cases), and maintained as the world and user needs evolve. Chapter 6 looks at these issues.

There are many aspects of *safety*. NLG systems should not generate texts which include profanity, racist language, or other socially unacceptable language. They should not encourage users to do harmful activities such as self-harm, suicide, or killing someone [14]; they should not give misleading advice or information which could harm the user more generally (including poor medical advice [23]); they should not make the user feel stressed or depressed [12]; and they should not divulge sensitive data to unauthorised users. Furthermore, such behaviour should *never* happen. In other words, we should design NLG systems that never exhibit the above behaviour under any circumstances. This is not only a very important but also very difficult challenge, especially for NLG systems which use neural models (whether trained, fine-tuned, or prompted).

Like other software systems, NLG systems need to be *tested* before they are deployed and released to customers, in order to ensure that the system is robust and that generated texts are of acceptable quality. Of course classic software engineering techniques can be used for NLG testing, but NLG has the additional challenge that there are usually many ways to express information in words, so it is difficult to create *test cases* that pair an input with a specific target output (because there may be other outputs which are just as good). A related issue is that many NLG systems are *stochastic*, which means that they can produce different outputs from the same input on different runs; this also makes it harder to use standard software testing techniques.

Last but not least, *maintaining* NLG systems can be challenging. Software engineering tells us that most of the life-cycle cost of a software system is in maintaining the software, including adapting it to changing data sources and user needs and fixing bugs. Unfortunately, very little is known about maintaining NLG software, especially neural systems. Maintaining rule-based NLG is probably similar to maintaining programmatic code, but maintaining a system that is trained on data offers additional challenges.

One such challenge is adapting neural NLG systems as the world changes (what is sometimes called *domain shift*). For example suppose a new medication is introduced which is very effective at treating cancer, and we want a medical NLG system to consider this medication in its recommendations. Since the medication is new, we will not have a corpus of training data to update the system. Strickland

[190] points out that such cases were very challenging for the IBM Watson question-answering system.

1.6 Use Cases and Applications

The final part of this book (Chap. 7) describes some NLG use cases and applications. The chapter first introduces general guidelines for successful applications and then discusses four specific domains: journalism, business intelligence, summarisation, and health.

NLG is being used in a huge range of different applications. It is not possible to make general statements that apply to all of these, except perhaps that the necessary input data must be available, but in most cases successful NLG applications have the following characteristics:

- *Scalability and configurability*: The NLG system is used to generate a large number of texts for a large number of users. This usually means that the system is configurable to some degree, so that users can customise it for their requirements, data sources, etc.
- *Accuracy and utility*: The NLG system reliably generates high-quality useful and accurate texts. Even a small number of low-quality outputs can reduce users' trust in the system.
- *Acceptability*: Users accept and want to use the NLG system; they do not, for example see it as a threat to their jobs.
- *Benefits*: The system must provide significant benefits to its users, which exceeds its cost, and offer a good return on investment. This is not always needed in the short term if investment capital is available, but it is crucial for long-term success.

Looking at specific application domains, NLG can be used in *journalism* to automatically write some types of articles in newspapers and other media. Figure 1.6 shows a simple example of this, which is a BBC news article about the 2019 UK general election in a specific constituency. This was generated automatically using technology from Arria (my company). Like most such generated articles, the BBC election reports were checked by human journalists before they were released to the public. The advantage of using NLG in this case was largely speed; old news is stale news, and using NLG to produce these articles meant they could be released the morning after the election, which would not have been possible if the articles had been manually written.

NLG can also be used to explain and summarise data in a *Business Intelligence (BI)* context, including data about sales, profits, inventories, customers, suppliers, employees, etc. At the time of writing, most BI data is communicated graphically using BI tools such as Tableau, but some types of insights are best communicated in words instead of data visualisations. BI NLG systems can be very scalable: Almost every business in the world is interested in sales, profits, etc.

Florence Eshalomi has been elected MP for Vauxhall, meaning that the Labour Party holds the seat with a decreased majority.

The new MP beat Liberal Democrat Sarah Lewis by 19,612 votes. This was fewer than Kate Hoey's 20,250-vote majority in the 2017 general election.

Sarah Bool of the Conservative Party came third and the Green Party's Jacqueline Bond came fourth.

Voter turnout was down by 3.5 percentage points since the last general election.

More than 56,000 people, 63.5% of those eligible to vote, went to polling stations across the area on Thursday, in the first December general election since 1923.

Three of the six candidates, Jacqueline Bond (Green), Andrew McGuinness (The Brexit Party) and Salah Faissal (independent) lost their £500 deposits after failing to win 5% of the vote.

This story about Vauxhall was created using some automation.

Fig. 1.6 Example story from BBC election reporter, from <https://www.bbc.co.uk/news/technology-50779761>

Another long-standing use case of NLG is summarising textual information. Note Generator (Sect. 1.3) summarised doctor-patient consultations; NLG systems can also summarise emails, legal documents, news articles, product reviews, and many other types of documents.

Finally (at least in this chapter), NLG can be used in *medicine and healthcare*. A large number of medical applications of NLG have been explored in the academic literature over the years, including:

- *Reporting*: tools that help clinicians create medical documents, such as Note Generator (Sect. 1.3).
- *Behaviour change*: tools to encourage healthier behaviour in smoking [165], diet [11], etc.
- *Patient information*: tools that explain medical information and patient data to patients [120].
- *Clinical decision support tools*: tools that help clinicians decide on appropriate medical interventions for patients [148].

The commercial NLG community has explored additional opportunities, such as business intelligence for health organisations.

Although there are many exciting opportunities in using NLG in healthcare, it can be difficult to deploy NLG (and indeed AI) solutions in medicine. This is partially because of understandable safety concerns (we cannot take the risk that an NLG system could damage the quality of care) and also partially because the huge diversity of the healthcare sector (especially if we look worldwide) makes it difficult to build scalable solutions. In other words, we can build NLG solutions that help one hospital, but it is difficult to build solutions that help hundreds of hospitals because there is little standardisation between hospitals in IT systems, equipment, clinical procedures, administrative processes, etc.

1.7 Ethics

NLG systems need to behave *ethically*. In other words, NLG systems need to act in a manner which is acceptable to society and does not harm users or third parties.

AI ethics is a very broad area, which of course goes well beyond NLG; I will just mention a few issues here. One is *accessibility across languages and communities*. In particular, much better NLG technology and resources are available in widely spoken world languages (such as English, German, and Mandarin) than in *under-resourced* languages such as Scottish Gaelic, Maltese, and isiZulu; this may encourage speakers of these languages to make more use of English. Also some communities (such as gypsies and travellers in the UK) are under-represented in the corpora used to train large language models; applications developed for such models may therefore not work well for members of these communities. From a technology perspective, it would be good to have more work on NLG in under-resourced languages and for under-represented communities.

Another issue is *bias*. For example an NLG system may generate texts that use male pronouns to refer to doctors and female pronouns to refer to nurses; this can support and reinforce gender stereotypes. Such biases can manifest in more subtle ways as well [39]. Unfortunately it can be difficult for developers to detect such issues, especially if they are not in the biased-against group. For this reason it is useful to get a diverse set of users, with different demographics and background, to check NLG texts for bias.

Unethical use cases can also be a problem, such as generating fake news or misinformation which is intended to stop people from voting. Of course working explicitly on such applications is unethical. A trickier case is developing technology which can be applied to both ethical and unethical use cases, such as techniques for generating persuasive texts. It is hard to give crisp guidelines about ‘dual-use’ technology, but I encourage developers to think about this issue and decide what they personally are comfortable with.

Theft of intellectual property can be a concern with neural NLG systems based on large language models. If an NLG system simply regurgitates (in whole or in part) what a person has written, then the human author needs to be acknowledged. If the original human-written text was not licensed for general reuse (e.g. with a Creative Commons³ license), then it can only be reused with permission from the human author, who may expect some compensation.

Another issue is *job losses*. Like other AI and IT technologies, NLG can lead to jobs being automated; for example we may need fewer weather forecasters if NLG is used to generate weather forecasts. Automation can also make jobs less interesting and rewarding; for example journalists may be told to check and fix up AI-written articles (Sect. 7.2), instead of writing articles themselves. Again there is no easy answer to this problem; if we had refused in the past to adapt technology

³ <https://creativecommons.org/>

that automated jobs, then we would still be living in an agrarian society where 90% of people worked on farms. As above, I encourage developers to think about this issue and decide what they are personally comfortable with.

Last but not least, NLG researchers and developers need to ensure that experiments used to evaluate NLG systems are ethical and cannot harm experimental subjects or third parties; this is discussed in Sect. 5.3.3.3.

There are grey areas. For example in general it is not ethical for NLG systems to lie. However an NLG system that informs relatives about the state of a sick baby may wish to be economical with the truth when communicating with an elderly great-grandmother who could have a heart attack if she hears bad news [46].

1.8 A Very Short History of NLG

1.8.1 Early History

Speculation about algorithmic and mechanical generation of texts dates back at least to the seventeenth century [146]. The earliest work on rule-based NLG as described in this book was done in the 1960s, in the context of building machine translation systems. Whereas modern MT systems use neural models to directly create a target-language output from a source-language input, early MT systems usually analysed the source language input into an intermediate structure, made changes to the structure to suit the target language, and then generated text in the target language from the intermediate structure. This last step was essentially NLG, and researchers working on this were the first researchers who attempted to dynamically construct sentences using natural language technologies [175].

The 1970s saw the first PhD theses on NLG which were not connected to Machine Translation. For example Goldmans' PhD thesis developed techniques for generating texts from conceptual dependency models [73], and Davey's PhD looked at generating summaries of tic-tac-toe games [42].

The 1980s saw a lot more research activity in NLG, including the first work on data-to-text [103] (Fig. 7.4), and developments of text summarisation technology [125]. There was also a movement towards separating NLG into component tasks and in particular separating content decisions (*what to say*) from linguistic decisions (*how to say*). The NLG community also developed more of an identity, with the founding of the ACL Special Interest Group in Generation (SIGGEN). The first International Workshop on Natural Language Generation was held in 1983. Researchers also began looking at NLG applications, for example in weather forecasts [93].

1.8.2 1990–2014

Research interest in NLG grew in the 1990s, with researchers investigating a broad range of topics including generating texts in multiple languages, integrating text and graphics, and generating different texts for different users. The first shared tasks appeared in text summarisation, such as SUMMAC [122]. Software libraries for doing NLG tasks were released [54], so researchers did not need to build everything from scratch.

In the late 1990s, researchers started looking at data-driven techniques for NLG, including ngram models, statistical models, and machine learning (the systems mentioned above all used rules or algorithmic code). This was successful in text summarisation [153], but less successful in data-to-text systems.

Evaluation also grew in importance, and researchers experimented with a wide range of techniques, ranging from simple metrics such as BLEU [145] and ROUGE [115] to large-scale task-based evaluations with real users [123, 165]. The first meta-evaluation studies were done to see how well different types of evaluation agreed with each other [161].

From a commercial perspective, this period had an exciting start, when CoGen-Tex (the first-ever specialist NLG company) operationally deployed the FoG weather forecast system in 1992 [71] (Fig 1.7). However commercial NLG did not really begin to take off until the latter part of this period, when several specialist NLG companies were founded, including my company Data2Text (which was bought by Arria), Ax Semantics, and Narrative Science.

1.8.3 2015–2024

This period is very recent, so it is difficult to give a historical perspective, but around 2015 neural deep learning technology started being used in NLP and NLG. Researchers had in fact been exploring using neural techniques in NLG since the 1980s [104], but such work had limited impact before deep learning became popular. Deep learning neural approaches were very successful in academic circles, and by 2020 this technology dominated academic NLP conferences and journals. Conferences also grew in size; 320 papers were presented at the Association for Computational Linguistic conference in 2015, while over 1000 papers were presented at this conference in 2023. Perhaps because of this growth, many exciting new uses for language generation technology were explored, such as image captioning (image-to-text). There was also a strong and welcome trend towards making code and data sets publicly available for other researchers. In 2022 the ChatGPT system was introduced, and the research community started focusing more on using prompted LLMs to generate texts

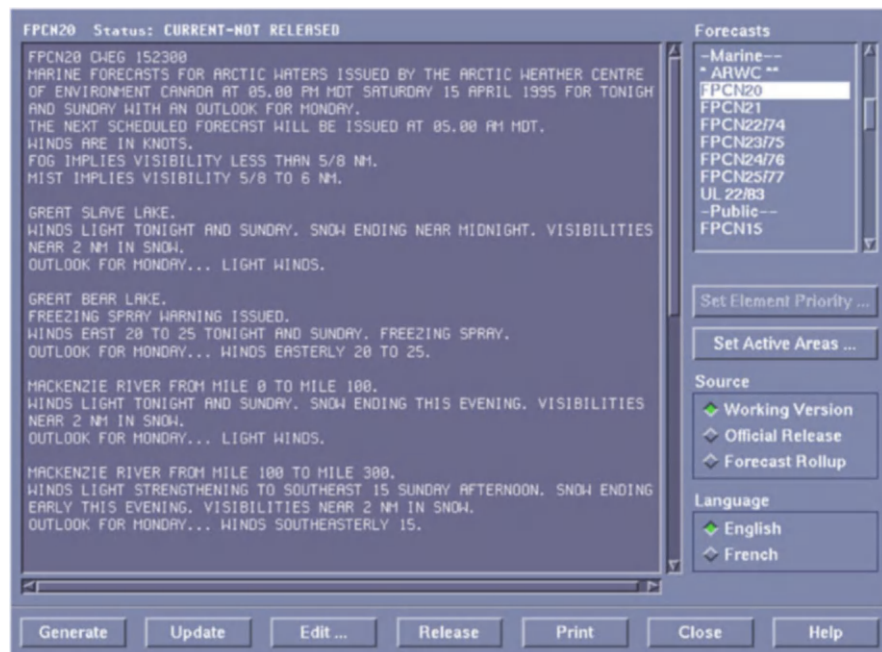


Fig. 1.7 FoG, the first-ever commercial NLG system, which went live in 1992. From cogen-tex.com (website is now defunct)

One consequence of the dominance of neural technology was that different types of NLG systems became more likely to use the same technology. For example the 2014 Arria weather system (Sect. 1.2) used completely different technology than 2022 Note Generator (Sect. 1.3). However both Note Generator and Facebook’s 2021 weather system [78] (Sect. 3.1.2) were based on fine-tuning and adapting the BART language model [113].

The commercial NLG world is changing extremely quickly at the time of writing. Companies initially were slower to adapt neural techniques than academics because of concerns over accuracy, quality assurance and testing, controllability, and maintainability. However, since the introduction of ChatGPT, there is huge interest in using prompted models to generate text in commercial applications. It is too soon to say where such models will prove successful in real-world usage and where concerns about accuracy, quality, etc. will motivate the continued use of rule-based or programmatic approaches

1.8.4 My Personal NLG Journey

On a personal note, I got my PhD thesis in 1990, from Harvard, on the topic *Generating Appropriate Natural Language Object Descriptions* [157]. The most important part of my thesis was work on generating *referring expressions* for entities. After my PhD, I further developed my ideas on this topic while doing a post-doc at the University of Edinburgh. I worked with Robert Dale, who was also interested in this area [41]. In 2000 Robert and I published the first-ever book dedicated to NLG, *Building Natural Language Generation Systems* [162].

In 1995, after spending a few years at CoGenTex (which developed the Fog system, Fig 1.7), I moved to the University of Aberdeen. Around 2000 I started focusing on data-to-text, working on projects such as SumTime (generating weather forecasts) [167] and Babytalk (generating summaries of electronic patient records) [148]. One of my papers, on the architecture of data-to-text systems [158], was given a Test-of-Time award by the INLG conference as the most influential paper published in any INLG conference before 2022.

In 2009 I founded a company, Data2Text, to commercialise data-to-text, and by 2012 I was spending almost all of my time at the company (which was bought by Arria in 2013). It was very exciting; we did a lot of work for the oil industry, and I really enjoyed working with engineers and exploring how NLG could help them.

Around 2017 I started to re-engage with academic life, and by 2023 I was once again a full-time academic. My research interests focused on evaluation, especially human evaluation (e.g. [18, 160, 195, 197, 222]); this partially arose from my frustration at the poor-quality evaluations that dominated early work in neural NLG, which meant that much of this work was scientifically meaningless. Evaluation quality has improved since 2017, but unfortunately many evaluations published in the academic literature are still of disappointing quality and/or difficult to reproduce.

1.9 Resources and Further Reading

Numerous surveys of NLG have been published in the academic literature, but most focus on technology and quickly become out-of-date. Likewise many textbooks on natural language processing include sections on NLG, but the focus is almost always on technology. Since technology evolves so rapidly, I do not give recommended sources here, but instead encourage readers to find up-to-date sources. There are also of course many commercial white papers on NLG, but these usually do not provide a balanced scientific view of the field.

Two surveys that go beyond technology are Gatt and Krahmer (2018) [65], who take a broad perspective on NLG, and Gehrmann et al. (2023) [68], who focus on NLG evaluation.

The Special Interest Group on Natural Language Generation (SIGGEN) (<https://siggen-acl.github.io/>) of the Association for Computational Linguistics (ACL)

organises academic conferences and other events about NLG topics, including the annual International Natural Language Generation (INLG) conference.

The *ACL Anthology* (<https://aclanthology.org/>) contains free downloadable copies of all papers published at conferences and journals sponsored by the Association for Computational Linguistics (and a number of non-ACL venues as well), including SIGGEN events such as INLG (<https://aclanthology.org/sigs/siggen/>). The Anthology is not always easy to search, but if you know what you are looking for, it is a great resource (and one that I personally use all the time).

Huggingface (<https://huggingface.co/>) is an excellent source for data sets and code libraries, especially for neural NLG. Popular Huggingface resources are often well documented and supported.

For further reading on topics discussed elsewhere in this book, please consult the relevant chapter. For example further reading on applications of NLG is available in Sect. 7.6.

The ACL maintains a good bibliography of papers specifically about ethics in Natural Language Processing, see <https://github.com/acl-org/ethics-reading-list>. The only paper I am aware of, which is specifically about ethics in NLG, is Smiley et al. [184].

Little has been published specifically about the history of NLG. However I gave a talk about this, which is available on YouTube at https://www.youtube.com/watch?v=SEw47Y_ZN8Q.

I have a website (<https://ehudreiter.com/>) which has many blogs about topics related to NLG.

Chapter 2

Rule-Based NLG



Natural Language Generation systems can be built using algorithms and rules which explicitly extract insights from data, structure information into narratives, and create good linguistic expressions of information; this is called *rule-based NLG*. Rule-based NLG is especially common in data-to-text NLG; it is rare in text-to-text NLG.

The advantage of rule-based approaches to NLG is that developers have complete control over what the system does, and the system is also testable and auditable. In other words, rule-based NLG is a ‘precision’ approach which lets developers build NLG systems with exactly the functionality which clients want and which (assuming no software bugs) will never go ‘off the rails’ and do crazy or inappropriate things.

The disadvantage of rule-based NLG is that writing (and debugging) the algorithms and rules can require a considerable amount of work, especially for NLG systems that generate complex texts. Of course good modularisation, structure, tools, and libraries will help (as with all kinds of software development), but writing rules for a large NLG system still can be a daunting task.

Even developers who do not use rule-based NLG will still benefit from understanding it, because it gives a good understanding of the sorts of processes and decisions which NLG systems must do. For example suppose an NLG developer uses neural language model technology (Chap. 3) to build an NLG system which summarises sensor data. The rule-based perspective shows that most such systems must detect and remove noise (Sect. 2.3). Unfortunately, many neural language models are not very good at removing noise from sensor data, so they will need help with this task (such as a separate preprocessor for noise filtering).

This kind of analysis is only possible if developers understand what an NLG system needs to do at a conceptual level. Such understanding also makes it easier to discuss requirements with users (Chap. 4) and to design appropriate evaluation schemes (Chap. 5).

2.1 NLG Pipeline

The simplest rule-based systems use *templates* to generate output texts from input data (Sect. 2.8); such functionality is provided by many packages, including Python’s Jinja library.¹ More sophisticated rule-based NLG systems structure the generation process into separate modules, which are often connected together into a *pipeline* (the modules run in sequence, for example the second module does not start until the first module has finished). In data-to-text, the most common modularisation is the data-to-text pipeline architecture [158], which divides the process into *Signal Analysis*, *Data Interpretation*, *Document Planning*, *Microplanning*, and *Surface Realisation* (Fig. 2.1).

Personal Note

I proposed this data-to-text NLG architecture in a 2007 paper [158], which was awarded a ‘Test of Time’ award in 2022 as one of the most influential NLG papers published in an INLG conference before 2022.

Not all systems include all of the above steps. For example we do not need to extract insights from data if they are directly present in the system’s input, and we do not need to design texts at a document level if the use case only requires a single sentence to be produced.

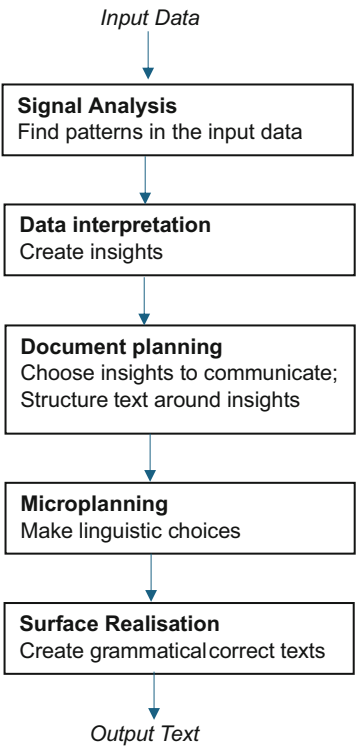
A variety of techniques can be used within the modules, including:

- *Algorithms*: We can use algorithms for many tasks, such as pattern detection within signal analysis.
- *Rules*: We can ask domain experts how they do a task and encode their response into the NLG system using rules. For example if we want to choose an appropriate verbal phrase (part of microplanning) to describe stock market changes (e.g. *inching up* or *skyrocketing*), we could ask experts (journalists and financial experts) how they choose verbs in this context and write rules based on this information.
- *Neural or other machine learning*: We can also use neural or ML approaches within modules (Sect. 3.2.6). For example for the verb choice task, we can build an ML model by analysing a corpus of historical financial reports [36].

The distinction between rules and algorithms is fuzzy. In theory algorithms define computational processes which can be reused in multiple NLG systems; they are usually implemented in programming languages such as C++, Python, or Java. Rules encode domain/genre/use case specific information which can only be used in this domain, genre, and use case; they can be defined in special rule

¹ <https://jinja.palletsprojects.com/>

Fig. 2.1 The data-to-text
NLG pipeline



languages, database or spreadsheet rows, or scripting languages. In practice it is often difficult to clearly separate rules and algorithms, especially since algorithms can be parametrised in a rule-like fashion for specific domains, genres, and use cases. Also rules can invoke algorithmic processing to make decisions.

In this book I will use ‘rule-based NLG’ to cover both rules and algorithms (which is what most academic researchers do) and not explicitly differentiate between these. The key issue is that both rules and algorithms require writing code to do NLG tasks and as such are very different from the data-based approaches described in Chap. 3.

2.2 Examples

In this section I introduce two example rule-based NLG systems, which I will use throughout this chapter to illustrate rule-based NLG.

Table 2.1 Example
DrivingFeedback data. Speed
limit is 30 on both King
Street and St Machar Drive

Time	Speed	Street
9:00:00	30	King Street
9:00:01	32	King Street
9:00:02	35	King Street
9:00:03	30	King Street
9:00:04	33	King Street
9:00:05	32	King Street
9:00:06	27	King Street
9:00:07	25	King Street
9:00:08	30	King Street
9:00:09	33	King Street
9:00:10	33	King Street
9:00:11	30	King Street
9:00:12	25	King Street
9:00:13	20	King Street
9:00:14	15	King Street
9:00:15	15	St Machar Drive
9:00:16	25	St Machar Drive
9:00:17	30	St Machar Drive
9:00:18	32	St Machar Drive
9:00:19	32	St Machar Drive
9:00:20	30	St Machar Drive

2.2.1 *DrivingFeedback*

The first example is a highly simplified version of the SaferDriver (Fig. 4.7) system [28]. The *DrivingFeedback* system takes as input GPS-type data showing the speed of a vehicle at regular timestamps, together with street name and speed limit information (which is extracted from the GPS data using reverse geocoding). The system produces regular reports on unsafe driving from this data. Table 2.1 shows example input data, and Fig. 2.2 shows how this data is processed by the different stages of the data-to-text pipeline, in order to produce the sentence *You sped twice on King Street* (a full report is several paragraphs long).

At a high level, *DrivingFeedback* works as follows:

- *Signal analysis* identifies unsafe driving incidents, such as speeding.
- *Data interpretation* clusters related unsafe driving incidents, such as multiple speeding incidents on the same road.
- *Document planning* selects most important clusters from a safety perspective and also organises content so that the report starts with positive feedback (i.e. congratulations on improvements from previous report) if possible, since this encourages people to read the report and take it seriously.
- *Microplanning* varies wording so that reports read differently from previous reports even if they have similar content.

Output of signal analysis (find speeding segments)

- Speeding(9:00:01-9:00:05, King Street, maxSpeed=35)
- Speeding(9:00:09-9:00:10, King Street, maxSpeed=33)
- Speeding(9:00:18-9:00:19, St Machar Drive, maxSpeed=32)

Output of data interpretation (cluster related speeding segments)

- SpeedingCluster(9:00:01-9:00:10, King Street, maxSpeed=35, incidents=2)
- Speeding(9:00:18-9:00:19, St Machar Drive, 32)

Output of document planning (choose which insights to include)

- Sentence: SpeedingCluster(9:00:01-9:00:10, King Street, maxSpeed=35, incidents=2)

Output of microplanning (design sentences to express insights)

- Sentence: Subject=*you*, verb=*speed* (past tense), modifier=*twice*, location=*on King Street*

Output of surface realisation (create actual text)

- Sentence: *You sped twice on King Street.*

Fig. 2.2 Processing DrivingFeedback data through the data-to-text pipeline

- *Surface realisation* creates grammatically correct texts.

2.2.2 Babytalk

A more complex rule-based data-to-text system is *Babytalk*. Babytalk is actually a family of systems which generate texts about babies in a neonatal intensive care unit (examples are shown in Fig. 2.3):

- *BT45* generates summaries of recent activity, which are intended to help doctors and nurses make decisions about interventions [148].
- *BT-Nurse* generates shift handover reports for nurses who are starting a 12-hour shift; the report summarises what happened in the previous shift and also in earlier shifts (in case the nurse has not dealt with the baby before) [83].
- *BT-Family* generates daily reports for parents, so that they know the status of their baby [120].

All of these systems extract information from the hospital's electronic patient record. At least at a conceptual level, they have a similar architecture but use different data interpretation and document planning rules and algorithms in order to select appropriate insights for the different use cases. BT-Family also uses a different set of linguistic expressions (microplanning) rules than BT45 or NT-Nurse, since it

Example BT45 output (extract):

By 11:00 the baby had been hand-bagged a number of times causing 2 successive bradycardias. She was successfully re-intubated after 2 attempts. The baby was sucked out twice. At 11:02 FIO2 was raised to 79%.

Example BtNurse output (extract):

Respiratory Support

Current Status

Currently, the baby is on CMV in 27 % O2. Vent RR is 55 breaths per minute. Pressures are 20/4 cms H2O. Tidal volume is 1.5.

SaO2 is variable within the acceptable range and there have been some desaturations.

Example BtFamily output (extract):

John was in intensive care. He was stable during the day and night. Since last week, his weight increased from 860 grams (1 lb 14 oz) to 1113 grams (2 lb 7 oz). He was nursed in an incubator.

Fig. 2.3 Example outputs from Babytalk systems

produces texts for non-specialists (parents) instead of domain experts (doctors and nurses).

At a high level, the Babytalk architecture works as follows:

- *Signal analysis* is done using standard tools, such as pattern detection and noise identification algorithms.
- *Data interpretation* is quite complex, in part because the system tries to detect and recover from some types of input errors (e.g. incorrect time stamps in the patient record). Much of this is done using *production rules*.
- *Document planning* is done using a combination of fixed *schemas* and dynamic narrative creation algorithms.
- *Microplanning* is done using algorithms for reference, aggregation, lexical choice, and other microplanning tasks.
- *Surface realisation* is done using the *Simplenlg* library [66].

Babytalk's processing is much more complex than DrivingFeedback's processing; at the time of writing, it is one of the most sophisticated rule-based NLG systems ever built.

2.3 Signal Analysis

The first stage of the data-to-text pipeline is *signal analysis*, that is finding patterns in the input data using signal processing techniques.

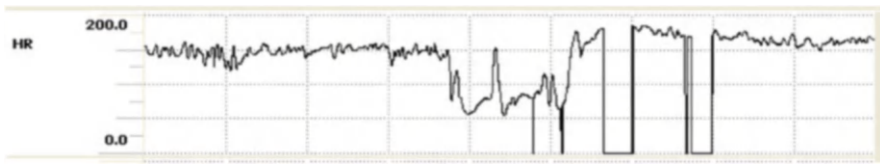


Fig. 2.4 Noisy data (HR, heart rate) from Babytalk. The data shows HR falling to 0, but the baby's heart did not actually stop beating

2.3.1 Noise Detection: Principles

Part of signal analysis is *noise detection and removal*; real-world sensor data is noisy, and hence sensor data may not be an accurate measurement of real-world events. For example the DrivingFeedback driving data in Fig. 2.1 shows a period of speeding from 9:00:01 to 9:00:05, except for a single data point at 9:00:03 where the speed drops to 30. Since GPS data is noisy (position measurements are not exact), DrivingFeedback treats this as noise and reports a single speeding segment from 9:00:01 to 9:00:05 (Fig. 2.2).

Babytalk systems got data from sensors attached to babies in a neonatal intensive care unit. When the baby kicked or was picked up by a nurse, the sensors would often show dramatic spikes or other changes which were purely due to the sensor momentarily losing contact with the baby's skin. An example is shown in Fig. 2.4; heart rate drops to zero at several points, and this is noise (the baby's heart did not stop beating). At any rate, these artefacts needed to be identified and removed so that Babytalk could focus on actual changes instead of sensor artefacts.

Personal Note

In the DrivingFeedback and Babytalk contexts, it at least is possible for noisy sensors to be replaced. I have worked on projects for oil companies where we used sensor data from sensors deep inside an oil well; it is not realistic to replace such sensors if they start misbehaving.

2.3.2 Pattern Detection: Principles

Once noise processing is done, the main signal analysis task starts, which is detecting patterns in the data. Which patterns are worth detecting depends on the domain. Statistics such as mean, range, and standard deviation are often useful. With time series data, in many use cases users want to know about spikes, trends, oscillations, and periods where the sensor data is outside of an acceptable range

(such as driving speed being above the speed limit). But there are also specialised patterns that are important in some domains but not others.

Pattern detectors in NLG systems must find patterns that human readers recognise. Many spike detectors for example use mathematical algorithms which trigger on things which do not look like spikes to human users; this is fine for most analytics, but not for NLG. In general, we want patterns that make sense to readers; finding such patterns is called *articulate analytics*.

One example of articulate analytics is when NLG systems use linear approximation to describe time series data. For example suppose we wanted to describe driving behaviour on St Machar Drive for the data in Fig. 2.1, for the period 9:00:15–9:00:18, where the driver is increasing speed. We could use:

- A simple *linear interpolation* and state the speed at the beginning and end of this period. An example is *After turning on to St Machar Drive, you speeded up from 15 to 32 mph.*
- A *linear regression* (trend line) to create a ‘best-fit’ segment to describe this driving behaviour, as shown in Fig. 2.5. An example is *After turning on to St Machar Drive, you speeded up from 17 to 33 mph.*

The linear regression is a better fit to the actual data and hence used by many analytics algorithms. However, in an NLG context, most readers find the regression text confusing and prefer the interpolation text [188]. Hence linear interpolation

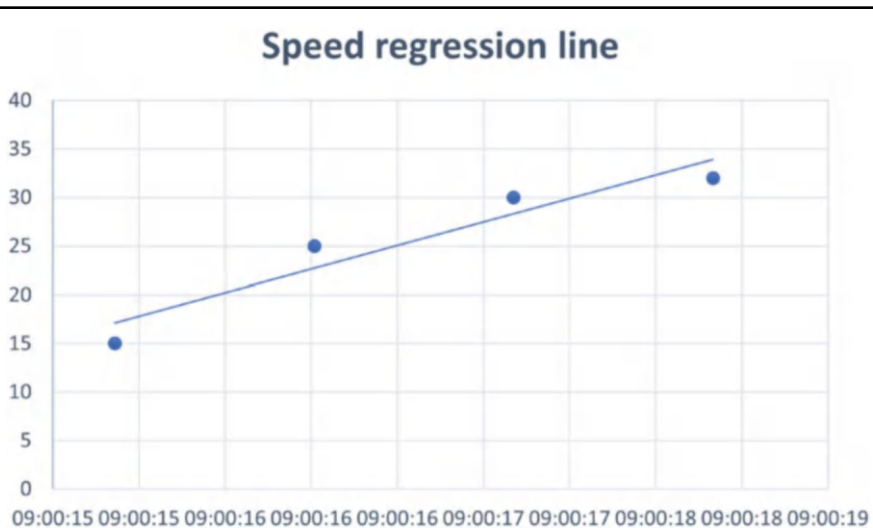


Fig. 2.5 Regression trend line for speed data. Initial value is 15 mph and final value is 32 mph. Regression line is 17 mph at the start of the period and 33 mph at the end. Readers prefer to see this data described as *you speeded up from 15 to 32 mph* (with actual values at beginning and end of period) instead of *you speeded up from 17 to 33 mph* (regression line)

should be used by data-to-text systems in such contexts, even though regression is more popular in many other analytical contexts.

2.3.3 *Techniques for Signal Analysis*

Signal analysis is usually done with standard noise suppression and pattern detection algorithms; the NLG developer generally selects what he or she thinks is the most appropriate algorithm from a standard pattern analysis library such as Python's SciPy and perhaps with some domain tuning.

For example Babytalk used fairly standard autoregressive modelling for noise detection, which was tuned on a clinical data set where artefacts had been manually annotated [83]. Babytalk used a variety of algorithms for pattern detection, including bottom-up segmentation for detecting trends [90].

2.4 Data Interpretation

The next stage of the data-to-text pipeline (after signal analysis) is *data interpretation*. The goal of data interpretation is to extract useful *insights* (sometimes called *messages*) from the patterns detected by signal analysis and indeed from the raw data in some cases. This step is of critical importance in data-to-text; an NLG system that presents useless or (even worse) incorrect insights is not going to be useful regardless of the quality of the language it generates. From a practical perspective, it is not at all unusual for NLG projects to devote more developer time to data interpretation than anything else.

2.4.1 *Principles*

The insights detected by data interpretation are quite varied but often can be characterised as one of the following:

- *Abstraction*: Combining patterns from signal analysis into a higher level insight, for example clustering together individual *Speeding* patterns into a *SpeedingCluster* insight, as shown in Fig. 2.2.
- *Interpretation*: Interpreting patterns. For example *DrivingFeedback* could interpret a *Speeding* pattern or *SpeedingCluster* insight as being *very dangerous* if the maximum speed exceeded the speed limit by 20 mph or more.
- *Linkage*: Relating events; this is very important for generating cohesive narrative instead of a bullet list of insights. For example if a driver did some speeding at the beginning of a trip and afterwards did no speeding, the system could say

You speeded on King St, but afterwards drove safely. Here, *but* links the two insights *You speeded on King St* and *afterwards drove safely*.

- *Importance*: How important and relevant are insights to the user? In the example shown in Fig. 2.2, for example we might give more importance to `SpeedingCluster(9:00:01-9:00:10, King Street, maxspeed=35, incidents=2)` than to `Speeding(9:00:18-9:00:19, St Machar Drive, 32)`, since the cluster is over a longer time frame and has a higher maximum speed.

One especially important type of Linkage is *casual reasoning*, for example *the baby's heart rate increased because of a nappy change*. Causal reasoning is difficult, but if it can be done, adding causal links increases the quality and usefulness of generated narratives.

2.4.2 Techniques

Data interpretation is domain-dependent and is based on what insights users want to know. It is usually done by writing rules or algorithms in consultation with users and domain experts; these rules are based on domain knowledge and often use domain-specific data analytics and mining techniques.

Once an NLG system is running, users (and domain experts) often want to modify and tweak data interpretation logic; in *DrivingFeedback*, for example a user might tell the system to ignore minor speeding incidents (less than 5 mph over the speed limit). Some commercial NLG systems have user interfaces which allow users or domain experts to inspect and modify data interpretation logic.

It should be possible in many cases to learn data interpretation models using machine learning, and hopefully we will see more work on this in the future. Because domain experts want to understand interpretation rules (as mentioned above), it often makes sense in this context to use ML techniques that produce interpretable models (such as linear regression and decision trees) which can be inspected, checked, and updated by domain experts; non-interpretable models (such as neural models) are often less desirable, even if they perform well.

Data interpretation can help in dealing with noise, in contexts where noise can best be detected by rules and consistency checks instead of by pattern analysis algorithms. In *Babytalk's* data, for example timestamps of actions were often incorrect. When clinicians performed surgery or other interventions, a nurse entered this information into the patient record, along with the time that the procedure was done. However this was usually done after the surgery finished, and sometimes nurses did not remember the exact timing of the intervention, especially if the nurse had to look after the baby immediately afterwards and did not get around to updating the patient record until much later. Pilot studies suggested that human readers of *Babytalk* texts could be very confused by incorrect intervention times, especially

if this meant that intervention times did not link up with the sensor data (e.g. blood oxygen seemingly rose before the oxygen level in the baby’s incubator was raised, instead of afterwards). For this reason, Babytalk’s data interpretation module attempted to identify and fix incorrect timestamps, by looking for the expected sensor signature of the intervention in the sensor data.

2.5 Document Planning

The third stage of the pipeline is *Document Planning*. Its goal is to determine the content and structure of the generated text. Although in theory separable, in practice decisions about content and structure tend to be connected, which is why they are usually grouped together into this stage.

2.5.1 Principles

Document Planning is the interface between the analytics part of the pipeline (signal analysis and data interpretation) and the linguistics part of the pipeline (microplanning and surface realisation). Decisions about which content to include are related to calculations of importance (data interpretation), while decisions about document structure influence linguistic decisions such as where to place sentence and paragraph breaks (Sect. 2.6.3)

In the DrivingFeedback example we are using, the Document Planning module decides that the text should include the *SpeedingCluster* (King Street) insight but not the *Speeding* (St Machar Drive) pattern. This decision is primarily driven by importance (discussed above).

In some cases, it is useful to add unimportant insights because they improve narrative coherence. For example the Babytalk system gave higher importance to sudden changes in sensor data than to slower changes, since sudden changes are more significant clinically. The first version of the Babytalk document planner only selected high-importance insights, which led to texts such as

TcPO2 suddenly decreased to 8.1. SaO2 suddenly increased to 92. TcPO2 suddenly decreased to 9.3

Users complained that it made no sense for TcPO2 to decrease to 9.3 when it had previously decreased to 8.1. In this case, TcPO2 had risen between the events, but slowly, so the rise was not considered clinically important; hence it was not mentioned. But narrative coherence is better if we nevertheless mention the rise [163], for example by adding the phrase *After increasing to 19*:

TcPO2 suddenly decreased to 8.1. SaO2 suddenly increased to 92. After increasing to 19, TcPO2 suddenly decreased to 9.3.

```

procedure DocumentPlan(Insights, importance function, importance threshold)

    SelectedInsights = all Insights with importance above a threshold

    # we need at least one insight
    if SelectedInsights is empty, add the highest importance Insight

    # insights should be presented in time order in the text
    OrderedSelectedInsights = SelectedInsights ordered by start time of Insight

    return OrderedSelectedInsights

```

Fig. 2.6 Pseudocode for document-planning script for DrivingFeedback

2.5.2 Techniques

From an algorithmic perspective, most NLG systems use simple scripts or schemas for document planning. In other words, a script or piece of code defines the overall document structure and also specifies which insights should be included in the document at different points; of course the script can include conditionals, since the structure may depend on the insights produced by data interpretation. A simple DrivingFeedback example is shown in Fig. 2.6.

More complex approaches are possible. For example the Babytalk system uses scripts (in the above sense) for part of the documents it generates, but for other parts it uses a *key event* algorithm which selects a small number of key insights about events and then creates a paragraph around each of these events by finding linked insights, such as causes and consequences of the key event [148].

Attempts have been made to treat document planning as a reasoning task, where the system explicitly reasons about what information should be in the generated text [7]; however it is difficult to get this approach to work robustly in real-world contexts. Attempts have also been made to use psycholinguistic principles to inform document structure decisions [196], but it is not easy to translate psycholinguistic theories into workable NLG code.

Researchers are looking at using machine learning techniques to determine document structure [151], but document planning seems quite challenging for machine learning. In general neural NLG systems seem to be better at expressing content than at selecting content.

One reason for the continuing strength of the script/code approach is that it makes it relatively easy to modify document content and structure if users request this; changes are much harder with the other approaches mentioned above.

The output of the document planning process is a *document plan*. There is no agreed standard for representing document plans. Some developers use trees inspired by Rhetorical Structure Theory [124], but ad hoc data structures are more common.

Table 2.2 Different Microplanning choices for expressing buy(John,book,10)

Task	Version 1	Version 2
Lexical choice	John <i>bought</i> a book for £10	John <i>purchased</i> a book for £10
Reference	<i>John</i> bought a book for £10	<i>He</i> bought a book for £10
Aggregation	John bought a book for £10	John bought a book. It cost £10

2.6 Microplanning

The fourth stage of the NLG pipeline is *Microplanning*. Its goal is to decide how to linguistically express the selected insights and other messages, which words to use, what syntactic structures to use, etc. We can think of microplanning as tackling a number of conceptually discrete tasks, including *lexical choice*, *reference*, and *aggregation*; see example in Table 2.2.

2.6.1 Lexical Choice

Lexical choice is the task of choosing words to express concepts and insights. In the DrivingFeedback example shown in Table 2.2, for instance the microplanner decides to express incidents=2 using the word *twice*; it could also have chosen *on two occasions*, *on 2 occasions*, *several times*, etc.

2.6.1.1 Lexical Choice: Principles

The core lexical choice task is to map semantic content onto words. This is related to *lexical semantics* in linguistics.²

Sometimes lexical choice is straightforward, but in other cases challenges arise. One is that in many contexts it is important to vary which words are used, for example to express incidents=2 using *twice* in some cases but *on two occasions* in others. *Lexical variation* makes texts more interesting and less repetitive, especially in contexts where users receive regular reports (such as weekly driving feedback reports). We can also vary other linguistic choices, such as passive versus active voice, if this makes texts less repetitive to readers.

Another key issue is *individual variability*, i.e. different people use words in different ways; for example my daughter describes as *purple* a shirt that I would call *pink*. In the SumTime project on generating weather forecasts [167], we empirically explored how forecasts readers and writers used and interpreted time phrases such as *by evening*. We discovered that *by evening* was used by some

² https://en.wikipedia.org/wiki/Lexical_semantics

Table 2.3 Number of times five forecasters (F1, F2, F3, F4, and F5) used *by evening* to refer to different times in the SumTime weather forecast corpus. Most common usage by a forecaster is in **bold** font. F1 and F4 usually used *by evening* to refer to 1800, F3 usually used *by evening* to refer to 0000, and F2 and F5 were more varied in their usage [166]

Time	F1	F2	F3	F4	F5
1800	30	5	2	27	13
2100	13	6	8	2	11
0000	2	9	80	5	14
other	2	2	1	0	4

forecasters to mean 6PM and by others to mean midnight (Table 2.3). We also worked with forecast readers and discovered that a few readers thought the meaning of *by evening* depended on sunset time and hence season and/or on when people normally had their last meal of the day and hence culture.

Other researchers have also found differences in how individual users understand and use words. Berry et al. [22] looked at how people interpreted phrases such as *very common* which are used to describe risk and discovered considerable variation and also little agreement with usage recommended by official terminologies. Ramos Soto et al. [155] found differences in how people used and interpreted geographic terms.

2.6.1.2 Lexical Choice: Techniques

From an algorithmic perspective, in many cases developers can just write rules on how information is expressed. For instance, write a rule that the information `incidents=2` is expressed by the word *twice* (or varies between *twice* and *on two occasions*). Rules may depend on users. For example a medical system such as Babytalk may express a medical concept such as *Desaturation* as *desaturation* to a clinician, but *temporary drop in blood oxygen saturation* to a patient who is not familiar with medical terminology (although sometimes patients object to ‘dumbed-down’ language and prefer medical terminology).

Developers can also use machine learning to build models for word choice and selection. A number of researchers have investigated building models for verb choice in financial texts, such as *rose* or *soared*. Chen and Yao [36] analysed this task and essentially concluded that simple statistical models might be preferable to complex ML models.

ML and rules can be combined. For example in SumTime we essentially used ML to build an interpretable model to select time phrases; the data included an `author` feature. We inspected the model to identify words and phrases whose meaning was author-dependent (such as *by evening*, see Table 2.3) and then adapted the model (in consultation with a domain expert) so that it never chose such words. The result was a model which chose common words and phrases whose usage was stable across users (e.g. phrases such as *by midnight*, which everyone agreed meant 0000). User evaluations showed that this strategy worked well in selecting time phrases [167].

2.6.2 Generating Referring Expressions

A *referring expression* identifies an entity to the reader. For example I can be referred to as *Ehud Reiter*, *Professor Reiter*, and *him*, amongst many other forms. If a generated text refers to me, which of these forms should it use? In a sense, choosing referring expressions is a specialised form of lexical choice, but usually it has been treated as a distinct problem in the research literature.

2.6.2.1 Referring Expressions: Principles

Reference often depends on *context*, which linguists call *pragmatics*. Different referring expressions are appropriate in different contexts. For instance, in our DrivingFeedback example, *King Street* is a reference to a particular street. The referring expression can also provide information about the city, i.e. *King Street, Aberdeen*; this differentiates the street from *King Street, Dundee*. If the reader knows from context (previous sentences, associated map, etc.) that the text is referring to driving in Aberdeen, then it is fine to just say *King Street*. However if context does not give the city, then the text should say *King Street, Aberdeen* (or indeed *King Street, Aberdeen, UK* if context does not specify the country).

In general, referring expressions should uniquely identify their *target* in the current context. For example if I want to refer to the student Miriam Smith in a class where no other student is called Miriam, then I can just refer to her as *Miriam*. If however the class also contains student called Miriam Black, then *Miriam* on its own is ambiguous, and I should instead use the referring expression *Miriam Smith*.

Reference is complex, in part because there are so many different forms, not just reference to people but also to objects, times, geographic locations, abstract concepts, etc. Many of these specific types of reference use specialised words and linguistic constructs which do not work for other types of reference, such as *yesterday* for temporal reference. Also, in many cases the *initial reference* to an entity in a text uses a different form from *subsequent references*; for example *Professor Ehud Reiter* could be used as an initial reference to me, while *Reiter* is used subsequently.

2.6.2.2 Referring Expressions: Techniques

A variety of algorithms have been proposed in the research literature, especially for the referential task of choosing *definite descriptions* such as a *big black dog* to refer to a visually or otherwise salient entity [102]. Algorithms have also been proposed for pronoun usage [91] and for special types of reference such as referring to components of a complex machine [159]. The above-mentioned research uses algorithms and rules, but there is also work on using neural techniques for reference

[35]. van Deemter [45] presents the reference problem from a cognitive science perspective.

One problem with reference algorithms is that different referring expressions are used in different genres and contexts. For example pronouns are more common in informal writing than in legal documents and other types of very formal writing; this makes it difficult to propose universal pronoun selection algorithms. For this reason, commercial work on reference has put more emphasis on configurability and flexibility, so that developers can easily configure the types of referring expressions used in their documents [159].

Regardless of the specific algorithm used, the microplanner will need access to a model of relevant contextual information in order to choose referring expressions.

2.6.3 Aggregation

Aggregation is the task of packaging information into sentences. For example suppose the DrivingFeedback document planner had decided to communicate insights about speeding on St Machar Drive and speeding on King St (i.e. both of the Data Interpretation insights shown in Fig. 2.2). In this case, the information could be communicated in one sentence or in two sentences.

- You sped twice on King Street and once on St Machar Drive. (*one sentence*)
- You sped twice on King Street. You also sped on St Machar Drive. (*two sentences*)

2.6.3.1 Aggregation: Principles

In general, there are many ways of distributing information across sentences. Since complex sentences may be difficult for below-average readers to understand (including many non-native speakers), while simple sentences are understood both by below-average and above-average readers, most NLG systems opt to use a large number of simple sentences.

If related insights or messages are aggregated into the same sentence, sometimes *elision* can be used to reduce the length of the aggregated sentence. For example *Sales roles in Germany* and *Sales fell in France* can be aggregated into *Sales rose in Germany and fell in France* (instead of *Sales rose in Germany and sales fell in France*). Elision can make a useful contribution to shortening texts and indeed making them more readable.

One issue with aggregation is that it can lead readers to make inferences about how insights or messages are related to each other. For example in the Babytalk context, if the phrases *The nurse gave the baby morphine* and *The baby vomited* are aggregated into *The nurse gave the baby morphine and the baby vomited*, readers may infer a causal link, i.e. that the baby vomited *because*

of the morphine, which is misleading if something else (such as illness) could have caused the vomiting. Because of such concerns, Babytalk used a very cautious and conservative aggregation strategy.

2.6.3.2 Aggregation: Techniques

From an algorithmic perspective, Harbusch and Kempen [77] proposed a set of linguistically motivated algorithms for doing some types of aggregation and ellipsis; some of these are implemented within the *Simplenlg* package. Different kinds of aggregation are appropriate in different domains, as well as for different users, which suggests that aggregation techniques should be adapted to domains and users; adaptation can be done based on user feedback [207] and/or by learning from appropriate corpora [215].

From a commercial perspective, most systems that I am aware of restrict themselves to relatively simple aggregations within the microplanner, done by straightforward rules or scripts; this is partially because of the risk of unwanted inferences (such as morphine causing vomiting). There tends to be more emphasis on *conceptual aggregation*, usually done as a type of abstraction within data interpretation. For example generating an insight that sales rose in every quarter of 2020 by abstracting over individual insights that sales rose in Q1 2020, Q2 2020, Q3 2020, and Q4 2020.

2.7 Surface Realisation

A *surface realiser* generates actual texts in English or other languages, based on the linguistic decisions made in the microplanner.

2.7.1 Principles

Conceptually, the realiser takes care of grammatical details so that the rest of the NLG system does not need to worry about this. This includes:

- *Syntax*: For example forming the negated version of a sentence. In English, this requires adding *do* in some cases but not others; for example *I do not like you* (including *do*) but *I have not met you* (no *do*). The realiser takes care of *do*-insertion so that other parts of the NLG system do not need to worry about this.
- *Morphology*: For example forming the plural form of a noun. In English, this is usually done by adding *s* to the end of a word, but there are many exceptions; for instance the plural of *child* is *children*, not *childs*. Again the realiser takes

Fig. 2.7 Simplenlg example;
output is *Mary chased the
monkey*

```
SPhraseSpec p = nlgFactory.createClause();
p.setSubject("Mary");
p.setVerb("chase");
p.setFeature(Feature.TENSE, Tense.PAST);
p.setObject("the monkey");
String output2 = realiser.realiseSentence(p);
```

care of forming plurals, so the rest of the system does not need to worry about this.

- *Orthography*: For example in English . is usually added to the end of a sentence, but there are some exceptions, for instance we do not add a . if the sentence ends in an abbreviation that ends in . (*I went to Washington D. C.*, not *I went to Washington D. C.*) [141].

Additional types of grammatical processing are needed in some languages. For example an English surface realiser must also decide between *a* and *an* (Sect. 3.1.1); this is a *morphophonology* task.

The above examples are in English, but all naturally evolved languages have grammatical details which can be dealt with by a realiser.³ For instance in French we need to replace *de le* by *du*, in Mandarin we need to add classifiers to noun phrases in some cases [34], in German we need to deal with separable verbs [26], etc.

2.7.2 Techniques

A number of open-source software libraries have been created to do the above tasks in a fairly straightforward way, of which the best known (at the time of writing) is probably Simplenlg [66].⁴ Simplenlg started off as a Java-based English realiser but has subsequently been ported to Python and other programming languages and also adapted to work in many other human languages, including German [26], Mandarin [34], and Galician [32]. A simple example of Simplenlg is shown in Fig. 2.7. There are other packages with similar functionalities, such as pyrealb [107];⁵ a French example of pyrealb is shown in Fig. 2.8.

³ Artificially designed and constructed languages such as Esperanto have more logical and consistent grammars.

⁴ <https://github.com/simplenlg/simplenlg>

⁵ <https://pypi.org/project/pyrealb/>

Fig. 2.8 pyrealb example;
output is *Le petit chat*
sauta

```
loadFr()
print(S(
    NP(D("le"),N("chat"),A("petit")),
    VP(V("sauter").t("ps"))
).realize())
```

There are also open-source packages that just do morphology, such as the Python *inflect* package for English⁶ and Abed's Arabic language functions [2]. These are essentially *language functions* as defined in Sect. 2.8.2.

We can also use statistical and machine learning models in surface realisation. One approach is *over-generate and select*, where the realiser generates a number of different possible sentences (surface forms), using rule-based techniques, and then a statistical or ML model is used to choose the best of these [106]. The OpenCCG realiser library [216] includes this capability.

Large neural language models (Sect. 3.2.4) are very good at realisation and getting grammatical details correct. Such models can be used for surface realisation as defined in this section [132], but it is more common for them to be used for linguistic expression (i.e. including microplanning and realisation) or indeed for the entire generation task (Sect. 3.2.6).

From a commercial perspective, many fielded systems use relatively simple realisation processing, perhaps just morphology and simple syntactic processing [213]. An open-source toolkit designed to support such an approach is the Rosaenlg library.⁷

2.8 Template NLG

Instead of using a modularised pipeline, developers can use *templates* to generate texts in a single step from the input data. The template concept is a vague one and ranges from simple mail-merge templates to complex scripting languages such as Jinja which include conditional statements, arithmetic computations, and other programming constructs. I use it here to cover all approaches where rules or algorithms are used to generate texts in a single step without any decomposition into tasks such as data interpretation or microplanning.

⁶ <https://pypi.org/project/inflect/>

⁷ <https://rosaenlg.org/>

2.8.1 Principles

A simple example of a template is shown in Fig. 2.9. The input to the system is data about a sports match, including names and scores of teams; the output is a simple sentence saying who won the game. Writing this template is much simpler than building a complete modularised NLG pipeline.

Of course, real sports-writing applications probably need something more sophisticated. For example the system may want to vary the verb depending on the size of the victory, using *destroyed* for a victory of 20 points or more, *edged* for a victory of less than 5 points and *defeated* otherwise. This could be done using the template shown in Fig. 2.10.

There are many other desirable improvements, including:

- Varying words (Sect. 2.6.1.1), for example in some cases using *beat* instead of *defeated*

Simple template for sentence such as *The Washington Wizards defeated the Los Angeles Lakers, 111-95*.

```
if (teamA.score > teamB.score)
    'The [teamA.name] defeated the [teamB.name] [teamA.score] - [teamB.score].''
else if (teamB.score > teamA.score)
    'The [teamB.name] defeated the [teamA.name] [teamB.score] - [teamA.score].''
else
    'The [teamA.name] tied the [teamB.name] [teamA.score] - [teamB-score].''
```

Fig. 2.9 Simple template to produce a sentence describing the winner of a sporting match

Slightly more complex template for sentence such as *The Washington Wizards defeated the Los Angeles Lakers, 111-95*.

```
if (teamA.score >= teamB.score+20)
    'The [teamA.name] destroyed the [teamB.name] [teamA.score] - [teamB.score].''
else if (teamA.score >= teamB.score+5)
    'The [teamA.name] defeated the [teamB.name] [teamA.score] - [teamB.score].''
else if (teamA.score >= teamB.score)
    'The [teamA.name] edged the [teamB.name] [teamA.score] - [teamB.score].''
else if (teamB.score >= teamA.score+20)
    'The [teamB.name] destroyed the [teamA.name] [teamB.score] - [teamA.score].''
else if (teamB.score >= teamA.score+5)
    'The [teamB.name] defeated the [teamB.name] [teamB.score] - [teamA.score].''
else if (teamB.score >= teamA.score)
    'The [teamB.name] edged the [teamA.name] [teamB.score] - [teamA.score].''
else
    'The [teamA.name] tied the [teamB.name] [teamA.score] - [teamB-score].''
```

Fig. 2.10 Slightly more complex template to produce a sentence describing the winner of a sporting match

- Referring to teams just by city name (e.g. *Washington*) where contextually appropriate
- Adding interesting insights, such as winning streaks (*Washington has won its fifth match in a row*)
- etc.

In principle we can implement all of the above using a simple template structure such as the one in Fig. 2.10, but the code rapidly becomes very long and complex. At some point it will become easier to introduce modules or functions to choose the verb (*destroyed*, *defeated*, *beat*, *edged*), decide how to refer to a team, decide whether to mention a winning streak, etc. This can be done on an ad hoc basis, especially for student projects. In commercial contexts, though, structuring the NLG process into distinct modules (e.g. treating verb choice as a lexical choice process, as described in Sect. 2.6.1) is better because it makes it easier to reuse and maintain code. Dealing with edge cases and exceptional conditions is also easier with a structured modularised approach.

In short, templates make sense for simple projects with straightforward input data and indeed for demos that only need to work on a few examples. However, a structured approach (with modules and representations) works better for complex projects, especially in real-world commercial contexts. One exception to this is that domain experts with limited background in programming and software development often find it easier to do even somewhat complex projects as templates. However such projects can be very difficult to support and maintain in real-world usage.

Personal Note

I have seen a number of commercial projects where demos and prototypes were built using templates, because this was the fastest way to build these, but the production system was built using an NLG pipeline, because this was easier to maintain.

2.8.2 Techniques

Very simple templates (such as the one shown in Fig. 2.9) can be implemented using mail-merge in Microsoft Word and other word processing systems. In my experience, many domain experts like to use Microsoft Excel for templates, even though Excel is not intended or designed for this.

More complex templates are usually created with a scripting language, which may be embedded into a general purpose programming language; a good example is Jinja, which is embedded in Python. This approach makes it easy to treat template development as a programming task and in particular use functions within the template. In such cases the distinction between template systems and modularised

pipelined NLG systems can become hazy. In other words, the same processing is going on ‘under the hood’, and the issue is whether tasks such as lexical choice are handled by explicit lexical choice modules or by random functions which choose verbs and do other word choice tasks.

Many sophisticated template systems include *language functions* which perform some limited NLG processing. For example we could use the following template with language functions to produce the DrivingFeedback text *You sped twice on King Street*:

```
‘‘You sped [instancePhrase(NumInstance)] on
[streetRef(Street)].’’
```

In this template, `instancePhrase(NumInstance)` is a language function that returns *once*, *twice*, *three times*, etc. based on the value of `NumInstance`. Another language function is `StreetRef(Street)`, which returns a contextually appropriate referring expression for `Street`, such as *King Street* or *King Street, Aberdeen*.

Common types of language functions include:

- *Orthography*: For example `EnglishList(cow,sheep,pig)` could return *cow, sheep and pig*, taking care of edge cases such as using `;` instead of `,` when necessary.
- *Morphology*: For example `plural(child)` could return *children*.
- *Syntax*: For example `countNoun(2, child)` could return *2 children*.
- *Lexical choice*: For example `timePhrase(0000)` could return *midnight*
- *Referring expressions*: For example `personReference(EhudReiter)` could return *Professor Ehud Reiter*.

These functions can be integrated into the template system or accessed via generic Java or Python libraries, such as the Python `inflect` package.⁸

2.9 Further Reading and Resources

Gatt and Krahmer (2018) [65] survey (amongst other things) rule-based NLG as it stood in 2018. Since the NLG community has focused on ML and neural techniques in recent years, the Gatt and Krahmer survey is still a good source for work on rules-based NLG. My 2000 book [162] is dated, but some people still find that it is useful for understanding the basic concepts of rule-based NLG.

Signal analysis is essentially pattern detection and noise suppression, and there are numerous data science resources which can be used for it, such as the Python `numpy` and `scipy` libraries. Data interpretation is likewise related to data science

⁸ <https://pypi.org/project/inflect/>

and can draw on data science resources, although data interpretation for NLG can often be somewhat different from conventional data science.

Document planning is fundamental to rule-based NLG, but I am not aware of good surveys specifically about document planning. However the topic is covered in Gatt and Krahmer's survey [65] of NLG.

More resources are available for microplanning. Krahmer and van Deemter [102] survey work on generating referring expressions; this survey is a bit dated but can still be a useful source for fundamentals. van Deemter has written two books related to microplanning which are aimed at non-specialist audiences. van Deemter (2010) [202] discusses vagueness, including how this influences lexical choice in microplanning, and van Deemter (2016) [45] looks at reference.

Many open-source surface realisers are available, see Sect. 2.7.2. The `Simplenlg` package has a tutorial which explains its basics, and this can be a useful way to understand what realisers do and how to use them. There are also many template engines available, see Sect. 2.8.2. Unfortunately I cannot recommend any open-source document planning and microplanning libraries. Some realisation packages (including `Simplenlg`) do limited amounts of microplanning, and document planning is often done using scripting languages.

There are companies that specialise in building rule-based data-to-text NLG systems or systems that combine rules and machine learning; their websites can be useful sources to understand what is being used commercially. At the time of writing, some of the best known are Arria NLG (my company)⁹ and Ax Semantics.¹⁰

⁹ <https://www.arria.com/>

¹⁰ <https://en.ax-semantics.com/>

Chapter 3

Machine Learning and Neural NLG



Instead of building an NLG system using rules from domain experts, we can use machine learning (ML) techniques to create an NLG *model*. The model is trained on data (e.g. inputs and outputs for NLG) and can apply the behaviour it has learnt from the data to generate new NLG outputs from novel NLG inputs. At the time of writing, most machine learning in NLG uses *neural* models, that is models that are loosely inspired by how neurons in the human brain work.

There are many possibilities within the broad space of using ML and neural techniques in NLG. Perhaps the most fundamental distinction is in how models are trained (Fig. 3.1):

- **Trained models** are trained from scratch to perform the target NLG task; typically they need large amounts of task-specific training data.
- **Fine-tuned models** take a ‘pre-trained’ large language model (sometimes called a *foundation model*), which is usually trained on large amounts of Internet content, and adapt it for the target NLG task using a limited amount of task-specific training data.
- **Prompted models** directly use a pre-trained language model to perform a task, by giving the model a request (*prompt*); task-specific examples can be included, but this is not necessary.

Other important distinctions include whether the model does the complete NLG process or only part of it (for example generating texts from insights, with the insights produced by a separate component); and whether the model is used autonomously or whether a person manually checks its outputs (*human-in-the-loop*; this is discussed in Sect. 4.3.2).

Machine learning and neural technology for NLG has been changing very quickly over the past 10 years. N-gram models (Sect. 3.2.2) dominated until 2016 or so, early neural models such as LSTM (Sect. 3.2.3) then became the focus of attention until 2019 or so, transformers (Sect. 3.2.4) then became the most popular

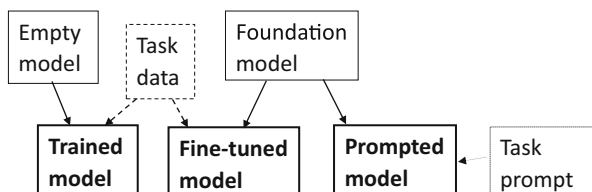


Fig. 3.1 Trained, fine-tuned, and prompted models

approach until 2022 or so, instruction-tuned models are pre-eminent at the time of writing. I would not be at all surprised if a new approach became prominent in 2025.

This book presents neural and ML technology at a very high level, focusing on fundamental concepts and issues which I believe will still be important in 2030. It deliberately does not attempt to give a detailed description of ML/neural technologies at the time of writing, since anything written along these lines in 2024 would probably be obsolete by 2025, and of only historical interest by 2030.

3.1 Examples

3.1.1 Very Simple Trained Model: *a* vs. *an*

One of the simplest ML models in NLG chooses whether (in English) to use *a* or *an*. This is difficult to do using rules, but easy to do using machine learning; this model also illustrates some of the fundamental concepts of ML in NLG.

In English, we use *a* before words that start with a consonant sound, but *an* before words that start with a vowel sound. Since this choice is based on how a word is pronounced, we cannot simply look at the first letter of the following word to make this choice; for example correct usage is *an umbrella* but *a university*.

Of course, there are many special cases, including the following:

- *Acronyms*: Pronunciation of acronyms depends on whether it is spelled out letter by letter; for example *an SAT prep course* if we spell out the letters *S-A-T*, but *a SAT prep course* if we pronounce *SAT* like the word *sat*.
- *Currency*: When we pronounce a currency, we say the number first even if the unit comes first in the written form. Thus we would write *an \$80 fine* because *\$80* is pronounced as *eighty dollars*, not *dollar eighty*, even though *\$* comes before *80* in the written form *\$80*.

Writing rules to choose between *a* or *an* is complex and difficult. A better approach is to analyse a large collection (*corpus*) of well-written English texts, such as Wikipedia to see whether *a* or *an* are preferred in front of specific words. For example *an umbrella* occurs 9500 times in Wikipedia, while *a umbrella* only

occurs 45 times; this tell us that *an umbrella* is the preferred usage. We can repeat this process for all words in Wikipedia and create a database which tells us when to use *an* and when to use *a*; this is accurate and does not require any rules!

This is technically called a *bigram* model, which means it is based on the frequency and probability of words pairs such as *an umbrella*. Bigram models are a type of *n-gram* model (Sect. 3.2.2). This is also a trained model in the sense of Fig. 3.1.

We can also *generalise* our findings. We know that the first few letters of a word are the most important in choosing between *a* and *an*, so we can look for cases where all words that start with the same letters have the same *a* vs. *an* behaviour. For example if all words starting with *d* use *a*, then we can replace the D-section of our database by a simple rule that any word starting with *d* takes *a* (this analysis can be automated). Not only does this reduce the size of our model, it also lets us handle new words which were not in the corpus. For example *dzild* is not in Wikipedia, but the generalisation process tells us to use *a dzild* instead of *an dzild*.

Of course, the above process only works if the corpus or *training data* is well-written English. Wikipedia satisfies this criteria, but Twitter does not. When machine learning models fail, it is often because of quality issues in the training data, and commercial ML models builders often spend more time on data issues (Sect. 3.3) than on algorithms.

Lastly, language changes, it is not static. For example Figure 3.2 shows the usage of *a historic* vs. *an historic* over time, in British English. This shows that while *a historic* is more common now, *an historic* was more common before 1985. This means that models should be trained on recent data; they may make mistakes (such as predict *an historic*) if they are trained on old data. More

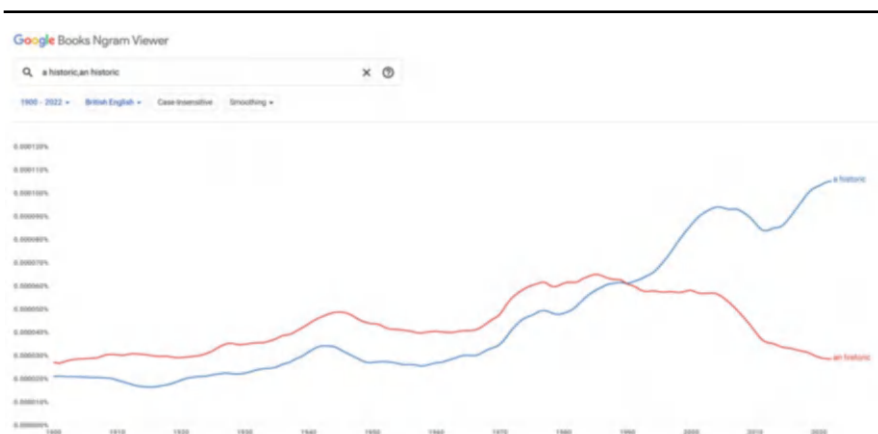


Fig. 3.2 Usage of *a historic* vs. *an historic* in British English, over time. From Google Books Ngram Viewer, <https://books.google.com/ngrams>

generally, we need to keep in mind that models trained on a corpus become less useful as the corpus ages; this is related to *domain shift* in machine learning (Sect. 3.4.1).

3.1.2 Fine-Tuned Neural Model: Facebook Weather Dialogues

An example of a more complex ML model is the NLG system developed by Facebook (Meta) to respond to weather inquiries [8], which was deployed and operationally used. This system relies on another component to select the content to be communicated in the text; it automates the task of generating a text from a meaning (content) representation. It uses neural technology and in particular relies on fine-tuning (Fig. 3.1) a large language model (BART [113]) to the task of generating weather information.

Figure 3.3 shows an example from this system. The Query is entered by the user. Other parts of the system do signal analysis, data interpretation, and some document planning and produce a *meaning representation* which specifies the content that should be communicated to the user. The NLG system then generates an actual text (similar to the reference text) which communicates the meaning to the user.

The meaning representation specifies the information to be communicated, for example that the low temperature will be 20 (`temp_low[20]`), and that it will rain on Sunday (`condition[rain] date_time[weekday[Sunday]]`). It also specifies connectives which relate insights (what I called *linkage* in Sect. 2.4.1), for example that the rain on Sunday contrasts with the sunshine on Saturday.

In order to fine-tune BART, the Facebook team needed high-quality examples of NLG inputs and outputs, that is of meaning representations and corresponding generated texts; this was the training data. The Facebook team used the following process to create these examples:

Query: How is the weather over the next weekend?

Response content (meaning representation produced by other components):

```
INFORM 1[temp_low[20] temp_high[45] date_time[colloquial[ next weekend ]]]
CONTRAST 1[
    INFORM 2[condition[ sun ] date_time[weekday[ Saturday ]]]
    INFORM 3[condition[ rain ] date_time[weekday[ Sunday ]]]
]
```

Reference (human-written): Next weekend expect a low of 20 and a high of 45. It will be sunny on Saturday but it'll rain on Sunday.

Fig. 3.3 Example input and reference (human-written) output for weather system, from [8]

- Asked engineers to create a varied set of queries and scenarios.
- Automatically generated meaning representations for these [9].
- Asked human annotators to manually produce high-quality responses from the meaning representation, following guidelines written by computational linguists.
- Had linguists check and verify the responses.

This process was used to create 25,000 query–response pairs for training models, and a further 6000 for testing and validation.

Note that the training data for this system was explicitly created for the project, and it was not scraped off the Internet. No cost figures are given in the paper, but creating 31,000 query–response pairs using the above process must have been a major task. The team subsequently developed techniques to reduce the amount of data needed [78].

At any rate, once the team had this data set, they then used it to fine-tune a BART model [113]. In other words, rather than creating a new model from scratch, they took an existing large language model and tuned it to work on the target NLG task (Sect. 3.2.4). The team also used knowledge distillation techniques (essentially training a smaller model on the fine-tuned BART model) to speed up run-time processing and reduce run-time memory requirements, which is very important in a production context.

Finally, because of the dangers of the neural model making mistakes, the team built a tool which identified generated texts which were possibly incorrect. They also implemented a backup rule-based NLG system which was invoked if the checking tool found potential problems in a text.

A key lesson from this example is that building a production NLG system based on a fine-tuned neural model required creating high-quality data sets for training [53] (Sect. 3.3), ensuring that models are acceptable from a time/memory/cost perspective, and implementing techniques to deal with model errors. The actual model training and tuning process (which is what most students and researchers focus on) is important, but its only part of the bigger picture if we are building real-world NLG systems.

3.1.3 Prompted Model: Using ChatGPT to Generate Weather Forecasts

If we want to build a system to generate weather forecasts using a trained or fine-tuned model, we need to collect a data set of thousands of high-quality input–output pairs, as was done for the Facebook system described above. An alternative approach is to take a large language model such as ChatGPT, which is trained on a huge amount of Internet material, and give the model a prompt (request) such as *Generate a short weather forecast for the below weather data*, followed by the numeric weather data (Fig. 3.4).

Input to prompted model (ChatGPT):

Generate a short weather forecast from the below data.

Weather data:

```
[
  {
    "time": 0,
    "wind": 10,
    "precipitation": 0,
    "temperature": 6
  },
  {
    "time": 3,
    "wind": 11,
    "precipitation": 0,
    "temperature": 6
  },
  ...
]
```

Output of prompted model:

Here's a short weather forecast based on the provided data:

Expect a day with changing wind speeds and temperatures. The wind will range from 9 km/h to 12 km/h throughout the day, occasionally gusting higher. The temperature will start at 6°C and gradually increase to 15°C in the afternoon before dropping to 10°C in the evening.

There is no precipitation expected, so it will be a dry day.

In summary, anticipate a mix of wind speeds, with temperatures ranging from 6°C to 15°C, and no rainfall.

Fig. 3.4 Example of using a prompted model (ChatGPT) to generate a weather forecast. The model is generic and has *not* been trained or fine-tuned to produce weather forecasts

If we want to control the wording and content of the weather forecast, we can explicitly request this in the prompt, or we can include a small number of examples (5 or 10, not 25,000) in the prompt (Sect. 3.3.5).

Prompted language models have many advantages over trained or fine-tuned models, including:

- No need to create a large set of domain-specific input–output examples.
- Since the model is general purpose, it is easy to add additional capabilities, such as generating texts in different languages.
- From a developer perspective, creating prompts does not require specialist expertise in machine learning, NLP, or programming (although these can help); fine-tuning or training neural models, in contrast, does require considerable expertise.

Many prompted models can be fine-tuned to improve their performance.

3.2 Machine Learning Models for NLG

Machine learning is a rapidly evolving field, and below I summarise a few of the different machine learning models and technologies which have been used to build NLG systems. This description is high level; Further Reading (Sect. 3.5) suggests additional sources for technologies.

3.2.1 *Classifiers*

A *classifier* puts an input data set into one of a finite number of *classes*. For example a sentiment analysis classifier classifies an input text into categories such as Positive, Neutral, or Negative. Classifiers are one of the oldest areas of machine learning, and indeed many classifier algorithms draw heavily on statistical research as well as AI. The `scikit-learn` Python library¹ includes a number of algorithms for building classifiers, including decision trees, Naive Bayes, K nearest neighbour, support vector, and many others.

To take a simple example, the SumTime weather forecast generator [167] needed to decide which verb to use when describing changes in the weather; this is an example of lexical choice (Sect. 2.6.1 discusses SumTime). In particular, in statements describing the wind, there is a choice between three types of verbs:

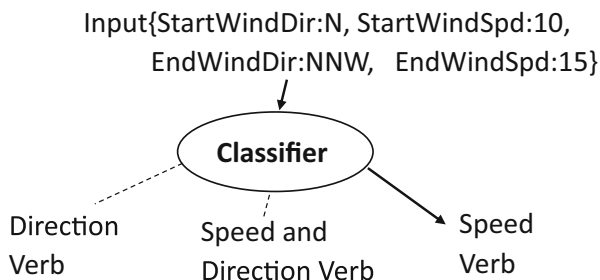
- Verb emphasising direction change, such as *W 10-14 veering N 12-16*
- Verb emphasising speed change, such as *W 10-14 increasing WNW 18-22*
- Conjoined verb emphasising both speed and direction change, such as *W 10-14 veering and increasing N 18-22*

SumTime used a classifier to decide on verb type (Fig. 3.5). Essentially, developers extracted all wind change phrases from a corpus of human-written forecasts and corresponding numerical weather data sets and used this to train a classifier which predicted the above choice from `<StartWindDirection, StartWindLowSpeed, EndWindDirection, EndWindLowSpeed>` tuples. SumTime used a decision tree, but other classifier techniques could have worked just as well [36].

Classifiers cannot generate texts on their own (i.e. they do not do *end-to-end* generation), but they are useful in making specific NLG choices, including content (insights) as well as language choices.

¹ <https://scikit-learn.org/>

Fig. 3.5 Classifier decides to use Speed Verb in a context where there is a large change in speed and a small change in direction



3.2.2 *N-Gram Language Models*

An n-gram language model is a model which gives information about the likelihood of a sequence of words. The *a-vs-an* algorithm (Sect. 3.1.1) is a simple example of this. Essentially given a word *XXX*, the model decides whether to use a *XXX* or an *XXX* by counting how often the *bigrams* a *XXX* and an *XXX* occur in a corpus (a collection of texts) such as Wikipedia. A bigram is a two-word sequence (2-gram); n-gram models support word sequences of other lengths including *unigrams* (one word) and *trigrams* (three words). Usually n-grams are structured to produce probabilities instead of raw frequencies, as can be seen in Fig. 3.2.

We can get good data on the frequency of bigrams and other small n-grams from corpora and use these to estimate probabilities. However, it is much harder to do this with large word sequences. For example suppose a speech-to-text system is trying to interpret an input sentence, and from an acoustic perspective is unsure whether the sentence was *I eat too much strawberry ice cream* or *I eat too much strawberry I scream* (since *ice cream* and *I scream* are very similar from an acoustic perspective). In theory it could decide between these based on frequency in a corpus. Unfortunately, the Internet (according to a Google search) contains *no* examples of either *I eat too much strawberry ice cream* or *I eat too much strawberry I scream*, so raw frequencies tell us nothing. However, we can use techniques such as Markov models and smoothing to estimate the probability of these word sequences from smaller n-grams such as *strawberry ice cream* and *strawberry I scream* [88]; this tells us that *I eat too much strawberry ice cream* is a much more likely sentence than *I eat too much strawberry I scream*.

From a software perspective, *nlk.lm* (part of the Natural Language Technologies toolkit²) provides good support for creating and using n-gram models.

N-gram models were popular for many years but at the time of writing have largely been replaced by neural models.

² <https://www.nltk.org/>

3.2.3 Early Neural Models

In recent years there has been huge interest in using *neural* (especially *deep learning* [75]) models to generate texts and indeed do other natural language processing tasks. These models are loosely inspired by neurons in the human brain. An enormous number of neural models and architectures have been discussed in the literature.

Neural NLG is usually considered to be a *sequence-to-sequence* task, where an input sequence (input texts for summarisation, time series for data-to-text) is converted into an output sequence of words (i.e., a text). Early neural models for NLG [72] used *recurrent* neural networks (RNNs), which are a type of neural network which iterates through arbitrary-length sequences of tokens (words). *Long short-term memory* (LSTM) architectures modify the core RNN to give the network ‘memory’ which makes it easier for the network to consider already-output words when it decides on the next word to output. *Encoder–decoder* architectures use two neural networks to generate texts, an *encoder* which maps the system’s input into an internal state, and a *decoder* which translates the internal state into generated texts.

These models were usually trained from data; that is, initially the model’s neural network is in a default state, and training algorithms such as *back propagation* are used to set the *parameters* (internal weights) of the neural network. Of course there are many variants of the above, and indeed many different possible neural architectures (the number of layers, the number of nodes in layers, etc.).

3.2.4 Transformers and Foundation Models

In 2017, Vaswani et al. [205] introduced the *Transformer* architecture. In very crude terms, while earlier neural models sequentially generated an output token from an input token (while using a memory mechanism to keep key information from earlier tokens), transformers could process larger chunks of information (such as complete sentences). They did this using an ‘attention’ mechanism which highlighted key connections between inputs to the model. For example when translating *I ate a grape* into (French) *j’ai mangé un raisin*, the attention mechanism tells the model that the word *grape* has an impact on how the word *a* is translated. Since *raisin* is a masculine noun in French, *a* is translated as *un*. In contrast, *une* would be used with a feminine noun such as *banane*, e.g. *j’ai mangé une banane*.

The transformer architecture worked very well in the context of fine-tuning large pre-trained language models, sometimes called *foundation* models. In other words, a large transformer model could be trained on a large amount of generic Internet content. This model could then be efficiently *fine-tuned* (adapted) to a specific domain and task with a much smaller domain/task corpus; this was done by updating the model’s parameters (links between nodes in the neural network) to better fit the domain/task corpus.

This had a huge impact, because it meant that developers could build powerful task-specific models with much less training data. This approach was used in the Facebook weather system described in Sect. 3.1.2 and also for the Note Generator system described in the Introduction chapter (Sect. 1.3). In both cases, it would not have been possible to acquire sufficient training data to allow high-performance models to be trained from scratch, but it was possible to get enough data to create high-performance models by fine-tuning a pre-trained model.

The transformer architecture also proved well suited to efficient execution on AI hardware and could be scaled up to very large language models (*LLMs*) with billions of parameters (links between nodes in the neural network) which were trained on tens or hundreds of billions of words. At the time of writing it is the dominant architecture for neural NLP and NLG systems and used in models such as BART [113], T5 [154], BLOOM [176], GPT [30], and PaLM [38]. Huggingface³ provides a comprehensive library and collection of open-source transformers [200] (and other neural models as well), supported by high-quality documentation and training material.

3.2.5 Instruction Tuning and RLHF

Researchers discovered that large language models such as GPT3 could perform many tasks without being adapted or fine-tuned, simply by giving them a *prompt* which made a request, presented the input data, and (optionally) included a few examples (Fig. 3.4) [30]. This was very exciting, since constructing data sets even for fine-tuning models could be a lot of work.

However, models such as GPT3 and BLOOM were very sensitive to the exact wording of the prompt. Using these models required a very good understanding of *prompt engineering*, i.e. of creating prompts which framed the desired task and use case in a way which resonated with the model and the data it was trained on.

Figure 3.6 shows an example using an early version of BLOOM. When we craft the prompt correctly, BLOOM gives us the translation (*rouge*) which we requested. However, if we use a different prompt which is intuitive but does not match corpus usage, then we get an explanation that it is impossible to have a single unique translation which works in all contexts. Which is true, but for most people the first response (*rouge*) is probably more useful.

One technique to make the models more robust and accept a wider variety of prompts is *instruction tuning* [211]. Essentially this process fine-tunes the model using a data set of common and intuitive instructions along with expected responses. The fine-tuned model learns how to respond appropriately to intuitive prompts; for example to respond to a request for a translation with the best possible translation, instead of explaining the difficulties of providing a translation.

³ <https://huggingface.co/>

Prompt:

Translate English to French.
English: red
French:

Response from BLOOM:

rouge

Prompt:

Translate English word "red" to French.

Response from BLOOM:

We, thus, conclude that there is no unique translation from English to French which preserves the various usages of the concept red, both in the sense of the noun color and in the sense of the adjective or adverb meaning "with red color".

Fig. 3.6 Effect of changing the prompt in an early version of BLOOM, which does not use instruction tuning or RLHF

Another technique for enabling models to respond appropriately to prompts is *Reinforcement Learning from Human Feedback* (RLHF). The core idea of RLHF is to collect a set of representative prompts, get the model to generate responses to these prompts, show the prompt–response pairs to humans, and ask the humans to assess whether the response was appropriate for the prompt. This feedback data is then used to update the model using reinforcement learning techniques [143].

Instruction tuning and RLHF are essentially forms of *alignment*, that is techniques which help models respond to human prompts in useful and appropriate manners which align with user expectations.

For instance, ChatGPT (which uses both instruction-tuning and RLHF) responds to the second prompt in Fig. 3.6 with *The translation of the English word ‘red’ to French is ‘rouge’.*; this is probably what most users are expecting.

Indeed, instruction tuning and RLHF in ChatGPT and other large language models worked so well that ordinary members of the public, who did not have a background in machine learning or natural language processing (or prompt engineering), were able to create prompts and use these models to generate texts and documents, answer questions, etc. Such models still gave better results with carefully engineered prompts (and indeed if they are fine-tuned for a specific task), but instruction tuning and RLHF meant that models could provide acceptable and useful responses ‘out-of-the-box’ in many cases from straightforward prompts. This had a huge impact, and when OpenAI launched a free version of ChatGPT, it reached 100 million active users two months after it was launched; nothing like this had been seen in AI before.

3.2.6 *End-to-end vs. Modular Architectures*

A final point is that neural and ML technology can be used in many different ways in NLG:

- *Component*: If an NLG system is organised as a set of components or modules (perhaps following the pipeline architecture presented in Sect. 2.1), then ML models can be used within specific components in order to perform specific tasks. One example is choosing *a* or *an* (Sect. 3.1.1); this is part of surface realisation (Sect. 2.7). There are many other examples, include detecting patterns in data, identifying the most important insights, and deciding whether to use pronouns. Castro Ferreira et al. [33] show how an entire NLG pipeline can be constructed from neural components.
- *Linguistic processing*: The Facebook weather system (Sect. 3.1.2) uses a neural model to do linguistic processing (convert insights into words) but uses other techniques to do data-side processing (choose insights to communicate). I have seen a number of systems that use this strategy, perhaps because *language* models by their nature are best suited to doing language processing.
- *Retrieval-augmented generation (RAG)* [114]: Another approach is to first use web search to find relevant information which may be useful to the user, and then give the search results, along with input data or prompt, to a neural language model. This reduces the amount of content determination which the model needs to do, and also gives it access to up-to-date information from the Web.
- *End-to-end* An end-to-end system uses an ML model to do the complete generation process; i.e. it takes in the system input data and produces output texts. This approach is popular with text summarisation systems, and also with simpler data-to-text tasks such as using ChatGPT to produce weather forecasts (Sect. 3.4).

3.3 Training Data

Building machine learning models requires training data, preferably large amounts of high-quality data, so acquiring good data sets for training is very important. Indeed, in many ML projects, more effort goes into data acquisition than anything else. Especially since building models (once we have acquired training data) is usually straightforward because of libraries such as scikit-learn, Huggingface Transformers, Keras, etc.

This section looks at some issues around training data. The focus is on training smaller models, or fine-tuning large language models. Creating large models (such as GPT or PaLM) requires much more training data (billions or even trillions of words) for the core model, plus additional data for instruction tuning and RLHF. However, building such models is a specialist and expensive exercise which is only done by a handful of companies and organisations.

Personal Note

As an academic, I have seen many student projects which use ML for NLP. Unfortunately I have often seen students grab an impressive-sounding data set from Kaggle or a shared task repository, spend a lot of time building ML models, and then realise that the data is flawed, so their model is garbage ('garbage in, garbage out'). I have also seen a lot of published academic papers which suffer from the same problem. Do not make this mistake—carefully investigate the data *before* you spend a lot of time on model building.

3.3.1 Data Sources

Sometimes developers building an NLG system can use an existing data set, but real-world applications often require new data sets. These can be acquired in several ways, including:

- *Real-world data and tests*: The ideal data source is a pre-existing data set which contains input data and high-quality human-written output texts. For example the Note Generator (Sect. 1.3) system, which summarised doctor–patient consultations, was built by fine-tuning the BART model on 10,000 consultations which had been manually summarised by doctors and entered into the patient's electronic health record. Similarly the word-choice models for weather forecasts described in Sect. 3.2.1 were trained on a corpus of 1045 forecasts written by human forecasters.
- *Crowdworkers*: If developers cannot find real-world data and/or texts, a common practice is to get workers on *crowdsourcing* sites such as Prolific⁴ and Mechanical Turk⁵ to write texts from real or synthetic data. For example fifty thousand restaurant descriptions were created by crowdworkers for the E2E corpus [52]. If crowdworkers are used, developers must design the process carefully in order to maximise quality; see discussion of using crowdworkers for evaluation in Sect. 5.3.2.3, where similar concerns arise.
- *Domain experts*: It may not be possible to use crowdworkers (who essentially are members of the general public) in specialised domains. In such cases, domain experts can be asked to write texts from data. For example the PriMock57 data set [144] contains summaries written by clinicians of 57 mocked doctor–patient consultations. Of course asking domain experts to write texts requires a lot more

⁴ <https://www.prolific.com/>

⁵ <https://www.mturk.com/>

time and money than asking crowdworkers to do this; this is one reason why PriMock57 is so much smaller than the above-mentioned data sets.

Each of these techniques has its own challenges. For example real-world human-written texts are sometimes not as high quality as we would hope; Thomson et al. [197] found many factual errors in journalist-written stories about basketball games. Crowdworkers are paid per task, so they make more money if they do tasks as quickly as possible, which means they may not be very careful when they write texts (Sect. 5.3.2.3). And domain experts are usually very knowledgeable about content, but sometimes do not write very well.

3.3.2 *Data Set Criteria*

In an ideal world, the training data set is (A) large, (B) high quality, and (C) representative. In NLG, we also usually want data sets to include system inputs as well as system outputs. Unfortunately, in the real world most training data sets do not meet these criteria. Poor-quality training data will result in poor-quality NLG systems, regardless of the sophistication of the ML technology used (again, ‘Garbage In, Garbage Out’). This is one reason why most commercial system builders spend more time worrying about data than about algorithms or model types.

Size Training a good neural model from scratch requires a lot of data. The amount depends on the complexity of the NLG task and the size of the generated texts, but even a simple task like the E2E challenge [52] (generating 10-word restaurant descriptions from eight features of the restaurant) required a training data set of 50K (feature-set, description) pairs. However, less training data is needed to fine-tune a pre-trained model. For example about 25K of (meaning, output) pairs were required to fine-tune BART for the Facebook weather system (Sect. 3.1.2); these texts were substantially more complex than E2E, but data requirements were reduced because a pre-trained model was used. Indeed a fine-tuned model for this task could have been created with a few hundred elements [78] although output quality would have been lower.

Quality Training data also needs to be high quality. If developers simply scrape data off the Internet or ask random crowdworkers to generate data, then it is likely that the data sets will have quality issues (readers should also be very careful with data sets on data science repositories such as Kaggle, some are fine but many have quality problems). One approach is to create a bespoke workflow for creating high-quality data, such as the one described in Sect. 3.1.2. Another approach is to try to automatically clean data sets (i.e. identify problems and then either fix them or delete suspect data) [53]. Regardless of the approach taken, it is important to assess and evaluate data quality issues and understand where there are problems.

Representative Last but not least, training data needs to be representative of real usage. Sometimes researchers use easy-to-obtain data which is not representative,

for example training dialogue systems on data sets which contain dialogues from TV shows, which are easy to get but completely unrepresentative of real-world dialogue. In medical NLP, a lot of research has used the MIMIC data set [86], which is a great resource but comes from a single hospital in Boston, and is not representative of hospitals in general. Building representative data sets is hard, but it makes a difference in real-world performance. At minimum developers need to understand and evaluate how representative their data is, and identify contexts where the data set does not have good coverage of real usage.

3.3.3 *Impact of Training Data on Prompted Models*

Prompted models do not need application-specific training data (except perhaps for a handful of examples, Sect. 3.3.5), but they are impacted by data quality issues in the data which was used to train the underlying language model. At the time of writing, models used for prompting are primarily trained on Internet data (such as that gathered by Common Crawl⁶), since this is the only data source which is large enough to provide sufficient training data. However, Internet data has many problems, which possibly may impact the quality of text produced by the prompted model. These include:

- *Inappropriate content:* The Internet contains material which we do not want to see included in system outputs, such as pornography, racist and sexist rants, stereotypes, encouragement to commit crimes, etc. This is an aspect of *safety*, which is discussed in Sect. 6.1. One approach to this problem is to try to exclude such material either from the training data or from prompt responses; however, this is not straightforward [214].
- *Spam and malicious content:* As prompted models such as ChatGPT become widely used in society, it seems likely that spammers and other ‘adversarial’ agents will try to create Internet content which, when used to train the model, will guide the model towards producing a response that advances the spammer’s agenda. Spam of course is already a huge problem with Internet search, but the black box nature of neural networks may make it harder to detect spam in neural models.
- *Biased content:* Most Internet material comes from well-off individuals in rich countries, which introduces biases. Very little, for example, comes from homeless people, which means that if a model is generating texts about homeless people, it will primarily be driven by Internet content created by well-off people commenting on homelessness, not content created by homeless people themselves.

⁶ <https://commoncrawl.org/>

- *Low-quality content:* A related issue is that a lot of material on the Internet is not of high quality. For example Balloccu et al. [12] (Sect. 6.1.1.4) looked at using ChatGPT to give health advice and discovered that a lot of the advice was inappropriate, in part because it seemed to largely be based on comments on discussion forums (such as Reddit) instead of high-quality advice from health professionals [10] (see also [4]). This is probably because there is more data in discussion forums than in high-quality curated health websites.

Most of the above problems are worse when the model is asked to answer questions (Sect. 3.4.2), which is how many people use ChatGPT. However, even in use cases where the task is summarising or explaining input data (which is what this book focuses on), there is still potential for the above problems to lead to toxic content, spam, bias, etc. (Sect. 6.1.1.1).

Data issues also effect testing and evaluation data, as well as training data; this is discussed in Sect. 5.2.8.

3.3.4 Synthetic Data and Data Augmentation

If a sufficiently large data set cannot be constructed from real data, developers may consider *augmenting* the data set by adding *synthetic data* to it [57]. To take a simple example, assume a developer is training a model to generate weather forecasts and has 500 genuine human-written forecasts but needs 1000 forecasts in order to train or fine-tune an NLG forecaster. The developer could use a machine translation tool to translate each forecast into German and then *back-translate* the German version of each story into a new English version. The back-translation process gives a *paraphrase* of the original story, which can added to the training data set (Fig. 3.7). Of course the process can be repeated using other languages (French, Chinese,

Original English text (from Figure 3.3):

Next weekend expect a low of 20 and a high of 45. It will be sunny on Saturday but it'll rain on Sunday.

Google Translate's German translation of above English text:

Am kommenden Wochenende wird ein Tiefstwert von 20 und ein Höchstwert von 45 erwartet. Am Samstag wird es sonnig sein, aber am Sonntag wird es regnen.

Google Translate's English translation of above German text (this is the back-translation):

A low of 20 and a high of 45 are expected next weekend. On Saturday it will be sunny, but on Sunday it will rain.

Fig. 3.7 Example of using back-translation to create a paraphrased version of a human text. This can be added to a data set if more data elements are needed and it is acceptable to use synthetic data

Finnish, etc.) if we want even more versions. There are many other techniques for data augmentation [57].

Data augmentation can be very useful in many cases and indeed is widely used. To take one random example, the BLEURT metric for assessing the quality of generated texts (described in Sect. 5.4.3.2) uses synthetic data (in part obtained by back-translation, as described above) to tune the model used to assess text quality; this enhances the robustness and accuracy of BLEURT.

Data augmentation is not always helpful, however. Anyone using data augmentation should carefully evaluate the result of using the augmented data set to see if adding synthetic data actually helps.

Also, evaluation should not purely be based on synthetic data. Sometimes it is acceptable to evaluate using a mix of real and synthetic data, but do not evaluate purely using synthetic data. If the goal is to evaluate real-world effectiveness, the evaluation must (at least in part) be based on real-world data.

3.3.5 Including Examples in Prompts

In all of the above examples of prompts, the prompts have basically made a request and provided input data. It is also possible to include examples in the prompt; this is called *few-shot* (prompts without examples are called *zero-shot*). The examples are not used to train the model in a classic machine learning sense, but they can tell the model what the user is looking for and guide it to producing an appropriate output.

For example the first half of Fig. 3.8 shows the result of a request to BLOOM to summarise sports information, without an example. The result is fine in abstract, but suppose that a shorter and more direct sentence is more appropriate in the target application. The second half of Fig. 3.8 shows how an example can be used to guide BLOOM into generating a short and direct summary sentence.

BLOOM does not use instruction tuning or RLHF. Examples can be less effective with models that use instruction tuning and RLHF, such as ChatGPT, but they still are useful in many cases.

3.4 Issues

Many issues arise when using machine learning and neural techniques. Some of the most important apply to all text generation systems (not just neural ones), so I discuss them at length in Part 2 of this book. In particular, concerns about safety, bias, and fairness are discussed in Sect. 6.1, and concerns about maintenance are discussed in Sect. 6.3.

In this section, I more briefly discuss several issues which primarily affect systems built with machine learning and neural techniques.

Prompt (no example):

Summarise the following text

Text: Aberdeen ended their long wait for a Scottish Premiership away win by beating Dundee United 3-1 at Tannadice.

Summary:

Response:

Dundee United was defeated 3 -1 by Aberdeen at their home ground, Tannadice.

Prompt (with example):

Summarise the following text

Text: Celtic kept their hopes of winning a domestic treble alive after beating rivals Rangers 1-0 in the Scottish Cup semifinal on Sunday thanks to a goal from Jota.

Summary: Celtic narrowly beat Rangers 1-0.

Text: Aberdeen ended their long wait for a Scottish Premiership away win by beating Dundee United 3-1 at Tannadice.

Summary:

Response:

Aberdeen beat Dundee United 3-1 at Tannadice.

Fig. 3.8 Result of prompt without and with example to BLOOM

3.4.1 Domain Shift

A fundamental problem with machine learning models is that they become out-of-date; this is called *domain shift* and is related to software maintenance (Sect. 6.3). A model trained in 2020 will reflect the world in 2020 and hence may not give appropriate answers in 2024. Strickland [190] points out that the IBM Watson question-answering system suffered because it could not give up-to-date information about medical interventions which included the latest findings, and we saw in Sect. 3.1.1 that English (and other human languages) is dynamic and changes over time. Braun and Matthes [27] show that GPT 3.5 can give incorrect legal assessments because it ignores changes in laws. In data-to-text, a related issue is that new data sources become available (e.g. new types of scanners in hospitals) and users expect such data to be used by the NLG system; this will not happen if the NLG system was trained on earlier data which did not include the new scanner.

Domain shift issues were especially prominent during the Covid-19 pandemic, when activities that had previously been fine and indeed encouraged (like going for a hike in the countryside, or meeting friends in pubs) became inappropriate and indeed sometimes illegal. In a data-to-text context, systems which reported on business data became less useful because the business world changed as some income streams collapsed but new ones (such as emergency support from government) emerged.

At the time of writing, considerable effort is going into updating commercial prompted models to try to keep them up-to-date. However most of this work is not published because it is commercially sensitive. There is also a lot of interest in techniques such as retrieval-augmented generation [114] which augment (out-of-date) models with (up-to-date) web search results.

3.4.2 Question Answering

As mentioned above, the focus of this book is on NLG which generates output texts based on some kind of input data. However, many (probably most) users of prompted models also use them as general tools to answer questions, provide information, and create documents. In such use cases, the models are creating content based on Internet data they were trained on, and they are not just describing and summarising run-time input data.

I regard this as a risky thing to do, because of the data issues mentioned above, especially as most models do not tell you where the data comes from. Incorrect and dated content are also concerns (which relates to domain shift, Sect. 3.4.1).

Personal Note

Several years ago a few websites described me as an *Israeli computer scientist*, which is not true (I am not Israeli). I eventually got these websites to update their material on me. However, in 2023 ChatGPT still described me as an *Israeli computer scientist*, presumably because it was trained on old versions of websites.

Models can also generate content which is accurate but still not appropriate (Sect. 6.1.1.4). For example MedPaLM [182] generates the following text in the context of giving health advice.

Ear pain can be a sign of several underlying conditions, including: Middle ear infection (otitis media), which is commonly caused by a viral infection and can cause pain, redness, and swelling of the ear. External ear infection (otitis externa), which is commonly caused by a bacterial infection and can cause pain, redness, and swelling of the ear. Earwax impaction, which can cause pain, a feeling of fullness in the ear, and decreased hearing. Ear pain can also be a sign of a more serious condition, such as a brain tumor or a stroke. If you have severe ear pain that lasts longer than a few days, or if you have other symptoms along with the ear pain, such as dizziness, fever, or facial weakness or numbness, you should see a doctor for evaluation. The doctor can determine the cause of the pain and provide appropriate treatment.

When I showed the above text to a doctor, she commented that she would never say this to a patient, because it states that ear pain can be a sign of brain tumour or stroke. This is true, but rare, and telling this to a patient is likely to cause a considerable

amount of anxiety and stress. Hence it is better to drop this statement, while keeping the statement that the patient should see a doctor if severe ear pain persists.

3.4.3 *Auditability and Controllability*

In many contexts clients want to control the behaviour of an NLG system, for example to ensure that it uses language compatible with a corporate brand, that it says nothing which raises legal concerns (e.g. racist language), and more generally that the system's output is appropriate and not misleading. There are also contexts where developers need to prove to a third party (such as a regulator) that an NLG system never engages in unacceptable behaviour.

Doing this with a neural NLG system is challenging. Developers can control the output by adjusting training data, fine-tuning data, or changing the prompt, but this does not always work. Even worse, proving that a neural network with hundreds of billions of parameters never engages in a behaviour is not currently possible. It is probably more realistic to add a checking tool as a post-processor, which identifies and deletes unsafe or inappropriate content (Sect. 6.1.2.3); but again this does not always work, and also high-quality checking can get complex and expensive to develop.

3.4.4 *Legal and Regulatory Issues*

As mentioned above, it seems likely that neural NLG systems which are used in healthcare and other regulated safety-critical areas will need regulatory approval, and this may be challenging, in part because of difficulties in rigorously testing these systems (Sect. 6.2). Provenance is also an issue for models (especially prompted models) trained on large chunks of the Internet, since much Internet material is incorrect or otherwise inappropriate [69].

Companies which use language models can be legally liable for mistakes made by the models. For example the airline Air Canada deployed a customer service chatbot which gave incorrect (hallucinated) information about some of the airline's fares. A court ruled that the airline had to honour what the chatbot told customers even if it was not in line with the airline's actual fares.⁷

Another important legal issue is copyright and intellectual property. Early language models were trained on Internet material, such as Wikipedia, which is available under a *Creative Commons* license which allows the material to be used in all sorts of ways, including training language models. However, larger prompted

⁷ <https://www.forbes.com/sites/marisagarcia/2024/02/19/what-air-canada-lost-in-remarkable-lying-ai-chatbot-case/>

models such as PaLM and GPT4 are primarily trained on material which does *not* have such a license; it is not possible to build up a sufficiently large training data set purely from material which has a Creative Commons (or similar) license. This has led to lawsuits, where content authors claim that material they authored and put on the Internet has been used for commercial purposes (building a language model) which they did not authorise, and for which they receive no compensation. The situation becomes especially difficult if content authors believe that they have lost contracts (or even jobs) to systems which were trained on material produced by these authors.

At the time of writing, the legal situation is unclear, especially since some countries are considering changing the laws which govern usage of Internet content. One possibility is that some countries will allow models which are trained on random Internet material and others will only allow models which are built on appropriately licensed (e.g., Creative Commons) material; this international disagreement would not be ideal, to put it mildly. It would be much better to have a worldwide convention on usage of Internet material, similar to the Berne convention which governs copyright law globally. However, creating international conventions is a slow process; for example the USA only joined the Berne convention in 1989, which is *102 years* after the convention was accepted in 1887 by European countries including France, Germany, and the UK.

3.5 Further Reading and Resources

At the time of writing, Jurafsky and Martin are preparing a third edition of their classic textbook, *Speech and Language Processing*, which will focus on machine learning algorithms and techniques used in natural language processing. This will be a superb resource when it is formally published.⁸

Many large technology companies publish high-quality white papers and research papers about neural approaches to NLG. I have been particularly impressed by Google's blogs and white papers (<https://blog.google/technology/ai/>) and research publications (<https://research.google/>). Readers should remember that companies do not say bad things about their products, so there may be biases in analyses and evaluations in Google papers about Google products. But the material from Google and other technology companies can be a great resource for learning about how the latest technologies work.

There are numerous companies which offer language models and neural NLG systems, with new ones seemingly announced every month (sometimes every week). I cannot recommend specific ones, since anything I say at the time of writing may no longer be true when this book is published.

⁸ Before publication, draft chapters are available at <https://web.stanford.edu/~jurafsky/slp3/>

Huggingface (<https://huggingface.co/>) is an outstanding source for downloadable models, software, and toolkits for neural NLG. NLTK (<https://www.nltk.org/>) is an excellent resource for earlier statistical and ML models used in NLP. Python's *scipy* library contains classifiers and other generic ML tools.

There is a large literature on data issues in NLP. Rogers [171] is a good overall position paper on the topic, and Bender and Friedman [19] argue for comprehensive data statements. Feng et al. [57] survey data augmentation techniques, and Lu et al. [116] survey synthetic data in AI generally. There are many companies and consultants who will assist with data acquisition, and I will not recommend any specific companies here.

Hu et al. [82] describe the real-world challenges of dealing with domain shift in AI models during the Covid-19 pandemic; the paper is not about NLP, but it does graphically illustrate the problem in a real-world setting. Ramponi and Plack [156] describe the problem from an NLP perspective, and some potential approaches to dealing with it.

I am not aware of many research papers on auditing AI or NLP models. The UK National Audit Office has a nice summary at <https://www.nao.org.uk/insights/how-to-audit-artificial-intelligence-models/>.

AI law, governance, and regulation (including auditing rules) are evolving extremely quickly at the time of writing. The EU Artificial Intelligence Act (<https://artificialintelligenceact.eu/>) will establish a legal framework for AI in Europe, and other countries (including the USA and UK) are also moving ahead with efforts to regulate AI. Legal case law is also rapidly evolving, including use of Internet material for training language models. Readers should check the up-to-date legal situation in their home country; anything said in this book about AI laws, governance, and regulation is likely to be out-of-date.

Chapter 4

Requirements



One of the most important questions in applied NLG is user requirements; what do real-world users and other stakeholders want NLG systems to do? AI systems are only useful if they do what users want them to do [189].

Of course requirements depend on the use case, which is discussed in Chap. 7. There is a large literature on requirements in the general software engineering literature, which NLG system builders should be familiar with.

At the simplest level, requirements analysis is about understanding what texts the NLG system should generate from different inputs in different contexts: what is their content, and how is this content expressed. The first step in this process is to explore high-level issues such as:

- *Quality Criteria* (Sects. 4.1 and 4.2): What aspects of quality are important to the users? Are they more interested in readability or in accuracy? Does worst-case performance matter as well as average case? How important are non-functional aspects such as time needed to generate a text? *Different quality criteria are important in different use cases and contexts*; for example accuracy is not important when generating fiction, but it is very important when generating texts that support medical decision-making.
- *Workflow* (Sect. 4.3): NLG systems can be used in many different *workflows*. In particular, they can be used fully automatically (with no human involvement); with a human checking and *post-editing* the NLG output texts before they are released; or to create draft material which human writers can use if they wish. Each of these workflows has associated requirements. For example, NLG systems used in a context with human checking and post-editing do not need to generate perfect texts, but they must be integrated into a user interface which makes human post-editing as easy as possible.
- *Text and Graphics* (Sect. 4.4): A key question from a requirements perspective is where users want to see information communicated in words, and where they would prefer that information be communicated visually, for example using

information graphics. Often the best strategy from the users' perspective is to use a mixture of text and graphics to communicate information.

Various methodologies can be used to understand the requirements of an NLG system. *User studies* (Sect. 4.5.1) are often a good way for developers to develop a high-level understanding of the above issues. A useful technique for understanding at a more detailed level what content the NLG system should communicate in different contexts, and how this should be expressed linguistically, is *manual corpus analysis* (Sect. 4.5.2), where a (relatively small) set of human-written *target texts* is manually annotated for content and aspects of linguistic style.

4.1 Quality Criteria: Texts

Many different *quality criteria* [16] can be used to assess the usefulness and appropriateness of an NLG system. Chapter 5 examines evaluation techniques, here the focus is on what some of these criteria are, and how their importance differs in different contexts. This section looks at text-level quality criteria, and Sect. 4.2 looks at system-level criteria.

Many quality criteria have been discussed in the literature [16, 81]. This book will focus on the criteria listed in Fig. 4.1 and illustrate these using the example shown in Fig. 4.2; this shows a simple weather forecast produced by ChatGPT from basic weather data (Table 4.1).

Personal Note

The quality criteria in Fig. 4.1 are the criteria which I personally have used most often in NLG projects. There are additional criteria which can be important in some use cases, such as emotional appropriateness (Sect. 6.1.1.4).

Key text-level quality criteria

Readability: Is a text easy to read and understand?

Accuracy: Is the information in a text accurate and correct?

Content: Does the text include the most important and relevant insights for the user?

Utility: Does the text help the user?

Key system-level quality criteria:

Non-functional: Does the system generate texts quickly and cheaply?

Variation: Is different wording used in different texts (if this is desirable)?

Average and worst case: Do *all* texts meet minimal expected quality criteria?

Fig. 4.1 Selected quality criteria for NLG

Output of ChatGPT:

Here’s a short weather forecast based on the provided data:

Expect a day with changing wind speeds and temperatures. The wind will range from 9 km/h to 12 km/h throughout the day, occasionally gusting higher. The temperature will start at 6°C and gradually increase to 15°C in the afternoon before dropping to 10°C in the evening.

There is no precipitation expected, so it will be a dry day.

In summary, anticipate a mix of wind speeds, with temperatures ranging from 6°C to 15°C, and no rainfall.

Fig. 4.2 Example text produced from weather data (Table 4.1)

Table 4.1 Example weather data for a specific location over one day, at 3-hourly periods

Time	Wind speed	Precipitation	Temperature
0	10	0	6
3	11	0	6
6	12	0	7
9	9	0	8
12	10	0	12
15	12	0	15
18	9	0	12
21	9	0	10

4.1.1 Readability and Fluency

One fundamental criterion for NLG systems is that their output texts are easy for people to read. In the research literature, this is sometimes called *fluency* or *clarity* as well as *readability*; unfortunately terminology in the research literature is not standardised [81]. From a theoretical perspective, we can distinguish between *readability* (users can quickly read a text and understand its content) and *fluency* (users subjectively think that a text is fluent and well written), but in this book we will treat these as being different aspects of an underlying assessment of linguistic quality.

For example, readers may judge that the text in Fig.4.2 is reasonable but not excellent from a readability and fluency perspective. Most readers can read and understand the text, but it could be shortened and simplified (which would improve reading time), for example by dropping *There is no precipitation expected* and simply saying that this will be a dry day.

An example of an incoherent text produced by an NLG system in the sports domain is [194]:

Markieff Morris also had a nice game off the bench, as he scored 20 points and swatted away late in the fourth quarter to give the Suns a commanding Game 1 loss to give the Suns a 118-0 record in the Eastern Conference's first playoff series with at least the Eastern Conference win in Game 5.

Poor readability is usually unacceptable; if the user cannot read the text, it is not useful. However, the importance of high readability (very well-written text) vs. medium readability (as in Fig. 4.2) depends on the use case. In particular, many texts targeted towards professionals, including financial reports, legal documents, and medical summaries, need to be understandable and have appropriate content (see below), but users may not insist on very high levels of readability. On the other hand, high readability can be important in texts intended for the general public, especially if the target audience includes people who are non-native speakers or have limited levels of literacy.

4.1.2 Accuracy

Another fundamental criterion for NLG systems is *accuracy*; is the information in the text factually correct? Again different names can be used, such as *fidelity*. Another way of expressing this criteria is minimising *hallucinations*, that is statements which are not correct.

At the time of writing, a lot of attention is being paid to accuracy problems in texts produced by large language models and other neural NLG systems. It is rare for such systems to generate incoherent texts with poor readability, but unfortunately they can still sometimes generate texts with serious accuracy problems.

If we look at Fig. 4.2, it is mostly correct, but there are some accuracy problems:

- The text says that wind speeds are in *km/h*, but this is not stated in the input. In fact, these speeds are in *mph*. Hallucinating units of measure is not good practice. If the system does not know the units of measure, it should not guess them, and instead simply say *the wind will range from 9 to 12* (etc.).
- The text states there will be *changing wind speeds*. In fact, a wind speed range of 9–12 over 24 hours is pretty stable.

A wider range of problems would probably be seen if the input data was noisy in the sense discussed in Sect. 2.3.1.

Accuracy can be formalised in many ways (Fig. 4.3). One issue is whether output texts should be accurate with regard to their *input data*, or accurate with regard to the *real world*. For example, the text in Fig. 4.2 states that temperatures are in Centigrade (°C); this happens to be true but is not stated in the input data. If we are assessing accuracy against input data, this would be regarded as an inappropriate

Types of accuracy:

- Accurate with respect to *input data* vs. accurate with respect to *real world*.
- *Semantic* (literal) accuracy vs. *pragmatic* (inferential) accuracy.
- *Critical* information correct vs *all* information correct.

Fig. 4.3 Types of accuracy

hallucination; however it would be fine if we assessed accuracy against the real world.

Sometimes the term *extrinsic hallucination* is used for cases where extra information is added to the text, beyond what is in the input data. *Intrinsic hallucinations* are statements that contradict the input data.

Another issue is *semantic* (literal) accuracy compared to *pragmatic* (inferential) accuracy. For example, in a sports context, assume that in a football match between Aberdeen and Dundee, Bojan Miovski scores 2 goals for Aberdeen and Glenn Middleton scores one goal for Dundee. Consider the statement:

Miovskoi scored two goals for Aberdeen, and Middleton scored one goal.

This is literally correct, but it implies that Middleton also played for Aberdeen, which is incorrect. Again we can define accuracy only based on semantic correctness, or we can define it to also include pragmatic correctness.

Of course, different accuracy errors will have different impacts, and sometimes it is useful to distinguish between *critical* and *non-critical* factual errors. For example, when summarising a doctor–patient consultation (Sect. 1.3) [138], it is essential to accurately record information which impacts clinical care. Therefore *the patient is vomiting* is a critical error if this statement is incorrect (i.e. the patient is not vomiting), since this may change decisions about appropriate medical interventions. However, accurately recording information about family members is less likely to impact clinical care; hence *the patient's wife is vomiting* may be considered a non-critical error if this statement is incorrect (i.e. the patient's wife is not vomiting). Of course the definition of critical and non-critical depends on the use case. More categories are possible; for example Freitag et al. [59] classify errors as *Major*, *Minor*, or *Neutral*.

The importance of accuracy depends on the use case (Chap. 7). It is of paramount importance in many medical applications, where critical errors in particular are unacceptable, and in general it is very important that texts be semantically correct and also pragmatically correct (not lead to false inferences). In legal contexts, semantic accuracy is very important, but pragmatic (inferential) accuracy is sometimes less critical. At the other extreme, accuracy is irrelevant for fiction-generation systems. Journalism is an interesting case, where gross inaccuracy is unacceptable, but limited factual errors may be tolerated in some cases; Thomson et al. [197] found 1.5 factual errors (on average) in human-written summaries of basketball games.

4.1.3 Content

In addition to being accurate, it is usually important for the content of a generated text to contain the key insights and information that the user needs to know. A related concept is minimising *omissions*, where key information is not present in a text.

Content and omissions are harder to pin down than readability and accuracy, in part because they depend on the use case and target audience. Readability and accuracy can be defined in a general manner that is somewhat domain-independent (although their importance depends on the use case). However we cannot specify what content is appropriate independent of the use case.

For example, if we look at the text in Fig. 4.2, it is probably fine from a content appropriateness perspective for a forecast aimed at the general public. However, it would not be appropriate for a forecast for offshore oil rigs, because it omits detailed information about wind speeds. Some oil-rig maintenance procedures can only be done if the wind speed is less than 10, and the Fig. 4.2 text does not specify when wind speeds are greater than or less than 10.

From a content perspective, the simplest situation is when all of the information in the system's input needs to be communicated in the generated text, and hence any input information which is not communicated is a content error. This was the case in the E2E challenge [52], for example, where the system's input was a small number of features (about a restaurant), and the generated texts were supposed to include all of these features. However this is unusual, and usually NLG systems produce texts which summarise their input data in some fashion. Certainly, a data-to-text system whose input data is large (thousands or even millions of numbers [201]) must summarise, and it cannot explicitly communicate this amount of information in a readable text of realistic size.

When a system does summarise its data, its texts have appropriate content when they include the key insights and messages for the target user and use case. Sometimes the key insights from a content perspective are the same as the ones which can give rise to critical accuracy errors (Sect. 4.1.2). For example if we know that *The patient is vomiting* is a critical error if it is incorrect, it is likely that omitting this statement (if it is true) will be a content error.

4.1.4 Utility

Perhaps the most important quality criterion, but also the hardest to define and measure, is utility. Is the generated text effective in helping the user or otherwise fulfilling its communicative goal? Utility of course is completely dependent on the use case and target audience, even more so than omissions and content quality.

For example, Gkatzia et al. [70] analyse and evaluate NLG weather forecasts for the specific task of helping outdoors ice cream vendors decide when to sell ice cream (Sect. 4.4.1). The most important weather parameter in this context is usually rain

(very few people buy ice cream from an outdoors vendor if it is raining); temperature can also play a role. The utility of the text in Fig. 4.2 can be assessed in this use case; it is probably pretty good, since the key information about rain and temperature is clearly stated. On the other hand, the utility of this forecast would probably be poor for offshore oil rigs (as discussed in Sect. 4.1.3).

Many NLG systems are deployed in *human-in-the-loop* workflows where a person checks and edits NLG texts before they are sent to end users (Sect. 4.3.2). In such cases, *post-edit time* (the amount of time that people need to check and edit the NLG texts) is often closely linked to utility; the value of the system is lower if people need to spend a lot of time checking its output texts. Of course, post-edit time depends on user interface and workflow as well as the generated texts. It is also variable across users, with some people doing considerably more edits than others [187].

We also need to keep in mind that real-world utility can depend on non-technical factors. For example, Babytalk BT-Family system (Sect. 2.2.2) generated summaries for parents of the status of a baby in a hospital neonatal ICU [120, 121]. The system was deployed and used by parents. When we asked them about utility, they said it would be much more useful if the summaries could be put on the Internet, especially since in many cases one parent was staying at home with other children, while the other parent was spending a lot of time with the sick baby in the hospital. Unfortunately, the hospital's IT security policy prohibited doing this. Hence BT-Family's utility was reduced by real-world factors which had nothing to do with NLG.

4.2 Quality Criteria: Systems

Most work on quality criteria has focused on the quality of texts produced by NLG systems, but there are also important criteria at the *system* level.

4.2.1 Non-functional Requirements

In many cases *non-functional* requirements are important, such as the amount of time required to produce a generated text. In interactive systems such as chatbots, the general rule is that the computer system should respond within one second [140], and sometimes this can be challenging for systems based on large neural language models.

For example, it took an early version of ChatGPT (GPT 3.5) around 1 second to produce the text shown in Fig. 4.2 (later versions are considerably faster). In an interactive chatbot context, 1 second may be acceptable (although it is not great); 5 seconds would not be acceptable.

A related issue is compute resources and costs. Running a large language model can require sizeable compute resources, and such resources may not be available

locally, for example if the NLG application needs to run on a phone in airplane mode (no Internet access). The situation is easier if the NLG application can access compute resources on the cloud, but such resources must be paid for, and there will be limits to acceptable expenditure. In late 2023 generating the Fig. 4.2 text using GPT4 cost around US\$0.01, which would probably be too expensive for a weather app that relies on advertising for income.

Privacy and security are also very important in many commercial contexts. In particular, some companies will not use cloud-based services for processing sensitive data because of privacy and security concerns and insist that all processing be done internally.

4.2.2 Consistency and Variation

In many use cases, users want texts to be *consistent* and always express information in the same way. In medicine, for example, texts describing clinical phenomena should be precise and use standard medical terminologies, in order to minimise the chances that the text is misunderstood. This is especially important when texts may be read by non-native speakers with variable levels of literacy [164].

However, there are also use cases where users want to see *variation* in texts, where the same information is expressed differently on different occasions. This is especially important in contexts where readers see many texts produced by the NLG system, such as journalism, sports reporting, and weather forecasts. Readers will get bored if they see the same language repeated in media stories [48]. Figure 4.4 shows some possible variations of the last sentence in Fig. 4.2, where the same content is expressed using:

- Different words (e.g. *no rainfall* vs. *dry*)
- Different ordering of phrases (e.g. start with wind speed information or start with precipitation information)
- Different semantic structures (e.g. whether reader is told something will happen, or told to expect that something will happen)

A few variations of a sentence in a weather forecast:

- *Original:* In summary, anticipate a mix of wind speeds, with temperatures ranging from 6°C to 15°C, and no rainfall
- *Variation 1:* In summary, expect a mix of wind speeds, no rainfall, and temperatures ranging from 6°C to 15°C.
- *Variation 2:* In short, the day should be dry with temperatures between 6°C to 15°C and a mix of wind speeds.

Fig. 4.4 Variations of the last sentence in Fig. 4.2; content is the same, but it is expressed differently

Sometimes users care about both consistency and variation; for example in a weather forecast context they may want consistency in content words such as *dry* but variation in other types of words such as connectives (e.g. alternating between *but* and *however*).

It is dangerous to generalise about use cases, but overall consistency is more likely to be important for NLG systems that generate texts for professionals, while variation is most likely to be important in systems that generate texts for consumers or the general public. Of course there are exceptions!

4.2.3 Average vs. Worst Case

Another system-level issue is the distribution of quality criteria across generated texts. Usually different texts generated by an NLG system have different individual ratings on quality criteria. For example, if we look at accuracy, perhaps a few texts will have no errors, some will have one error, some will have two errors, and a few will have three or more errors. If we want to compute quality criteria for *systems* instead of individual texts, then we have to decide how to map a distribution of text-level quality scores onto a single system-level quality score. In principle we could report the distribution of text-level scores, but this is rare.

If we want to report a single number which aggregates the distribution, the most common choice is the mean (average) value of the criteria across a set of texts; this is *average-case performance*. However, in some cases (especially when safety is important, see Sect. 6.1), it is essential that *all* texts generated by a system have acceptable quality; in such cases we report the minimum value of the criteria across the set of generated texts (*worst-case performance*). An example is shown in Table 4.2.

If average readability is most important in this use case, we should report the mean readability of the generated texts (8). However, if the key requirement is that all texts achieve a minimum readability level, then we should report the minimum readability score (4). In some use cases we may want to report both, for example if we want a high average readability score but also a guarantee that all texts achieve a minimum readability level (e.g. not be incomprehensible).

Table 4.2 Average- and worst-case ratings of a set of texts

Generated text	Readability (0–10)
Tomorrow should be dry with temperatures between 6°C and 15°C	9
Tomorrow will be a dry day, with moderate temperatures	9
Tomorrow will be a moderate day with dry temperatures	4
Tomorrow will be a really nice day	10
Average (mean) readability: 8	
Worst-case (minimum) readability: 4	

4.3 Workflow

Another key aspect of requirements is how the NLG system fits into overall *workflows*. In particular, it is useful to distinguish between *fully automatic NLG*, *NLG with human checking and post-editing*, and *humans creating a document from an NLG draft*.

A related concept is *integration*, that is connecting the NLG system to input data sources and output document delivery mechanisms. This is very important for production systems; in commercial NLG projects, a sizeable chunk of engineering effort is often spent on integration. However, since integration issues for NLG systems are similar to integration issues for other types of software systems, I will not discuss them here.

4.3.1 Fully Automatic NLG

A fully automatic NLG system produces a narrative with no human involvement (other than humans reading the generated narrative).

At the time of writing, there is limited use of fully automatic NLG systems in professional contexts, in part because of the fear that some of the NLG texts may be incorrect. In other words, with regard to quality criteria (Sect. 4.1), there is a fear that worst-case readability, accuracy, and/or content appropriateness will not be acceptable; i.e. in a few cases the generated texts will be difficult to read, inaccurate, or missing key information. In safety-critical or high-value contexts, this is not acceptable. Of course, there are use cases, perhaps especially in consumer-oriented contexts, where a small number of mistakes are perhaps acceptable, and fully automatic NLG systems can be used in such contexts.

The weather system [8] described in Sect. 3.1.2 is an example of such a system. While some specialised types of weather forecasts (such as aviation forecasts) are safety critical, forecasts for the general public usually do not kill people if they are wrong. Also, weather forecasts by their nature are approximate and sometimes incorrect because they are *forecasts* (the result of a numerical model which simulates how the atmosphere will change in order to predict weather parameters over time), so users know that they cannot be 100% trusted even ignoring NLG issues.

4.3.2 Human Checking and Editing

In NLG systems used by professionals (doctors, accountants, engineers, etc.), it is more common to see workflows where a person checks the output of the NLG system and fixes (*post-edits*) the text if necessary. This is certainly the case with systems that generate medical documents such as consultation summaries [95]

Original SumTime wind text: SW 20–25 backing SSW 28–33 by midday, then gradually increasing 34–39 by midnight.

Human-edited version of above text: SW 22–27 gradually increasing SSW 34–39.

Fig. 4.5 Example of post-editing a SumTime wind text, from [187]. Human has changed 20–25 to 22–27, dropped the phrase *backing SSW 28–33 by midday*, and also dropped *then* and *by midnight*

(Sect. 7.5.1), and it is also true of most NLG systems which are used to generate journalistic content [48] (Sect. 7.2).

In the weather domain, the forecasts produced by SumTime were checked and edited by human forecasters [187] before they were sent to clients. This is partially because these were specialist forecasts for people operating offshore oil rigs, so accuracy requirements were higher than for public forecasts (Sect. 1.2). Figure 4.5 shows an example of how a SumTime text describing the wind was edited by a human forecaster before being released to clients.

Sripada et al. [187] analysed a corpus of such edits to SumTime forecasts (real edits made during operational usage of the system) and discovered that there was a considerable difference in the amount of post-editing done by different forecasters. This is partially because some forecasters are fussier than others, but also because some forecasters insisted on editing texts into their personal preferred style. If consistency is important, editors should be given clear guidelines on what should be fixed, ideally supported by training.

Post-editing also of course is influenced by the editing user interface and by how it fits into the editor’s overall workflow [95]. A good post-edit UI can make post-editing much quicker and also can help editors conform to instructions.

The result of a workflow where human experts post-edit NLG outputs can be excellent. Moramarco [137] showed that for the task of writing summaries of doctor–patient consultations, texts produced by asking clinicians to post-edit summaries produced by Note Generator (Sect. 1.3) seemed to be slightly *more* accurate (on average) than texts produced by the same clinicians using a completely manual workflow.

We can also use machine learning techniques to learn from post-edits and refine our models so that they make fewer mistakes in the future. This is sometimes called *automatic post-editing* [31].

There has been extensive work on post-editing in machine translation. Indeed, there is even an ISO standard about this (ISO 18587:2017).

4.3.3 Creating Drafts for Human Writers

Finally, NLG systems can be used to generate draft documents which human writers can use when they create documents. For example an NLG system may generate a

financial report which is ‘mined’ for content by a human analyst; i.e. the human analyst does not try to fix the entire NLG narrative, instead he or she extracts useful chunks from the NLG narrative and includes them in the document he or she is writing. As with post-editing, different authors use drafts in different ways, and the user interface is very important.

The division between ‘human checking and post-editing’ and ‘creating drafts’ is fuzzy, but in general in the first case the NLG system is the main author, assisted by the human, whereas in the second case the human author is the main author, assisted by the NLG system. In principal, usage of NLG in a use case can evolve, perhaps starting with the system creating drafts, then moving to a ‘human checking’ workflow, and ending with fully automatic NLG with no human involvement.

4.4 Text and Graphics

Visualisations, images, and graphics can be used to communicate numerical data and in many use cases can be an alternative or supplement to texts produced by data-to-text NLG systems. For example:

- *Financial and medical information* can be presented using words, information graphics, or both.
- *Persuasive content* (e.g. to encourage safer driving) can be presented using words, graphics/images, or both.
- *Instructions* (such as directions for getting from A to B) can be presented using words, maps, or both.

Hence when building an app which communicates information to a user, developers may need to decide whether to use words (NLG), some form of graphic, or a combination of both.

Sometimes the choice is dictated by pragmatic reasons. For example, a voice assistant which communicates over a telephone landline cannot show images, so it must rely on words. Similarly systems for blind people emphasise words; it is possible in theory to communicate graphically using Braille displays, but this is expensive and usually avoided. On the other hand, use cases which require information to be understandable to speakers of many different languages (such as many DIY instructions) avoid words and use pictures instead. Summaries of textual documents are also usually done with words; pictorial summaries are possible but unusual.

In many use cases, however, we have a choice between words and pictures and hence need to decide which of these is most suitable for the use case and target audience; this is an important aspect of understanding client requirements.

4.4.1 Decision Support

One use of data-to-text NLG is *decision support*, that is presenting information to people to help them make decisions. A few studies have looked at the relative effectiveness of text and graphics in this area.

Gkatzia et al. [70] present an interesting study of this issue in the context of simple weather forecasts. Figure 4.6 shows an example of the kind of forecast they examined; this example shows both images and words. They also had an images-only condition (no words) and a words-only condition (no images). They ran an experiment where they showed people either the words+graphics forecasts, the graphics-only forecast, or the words-only forecast and asked them to make a weather-related decision. They found that the best decisions were made from words+graphics, and words-only forecasts were more effective (as decision aids) than graphics-only forecasts.

Gkatzia et al. looked at a context where ordinary people were making a decision based on fairly simple data. The Babytalk BT45 systems [148] generated texts which were intended to support clinical decision-making, and studies were done of the text-vs-graphics issue in this context, where medical professionals examine clinical data and then make decisions based on this data. Law et al. [109] compared

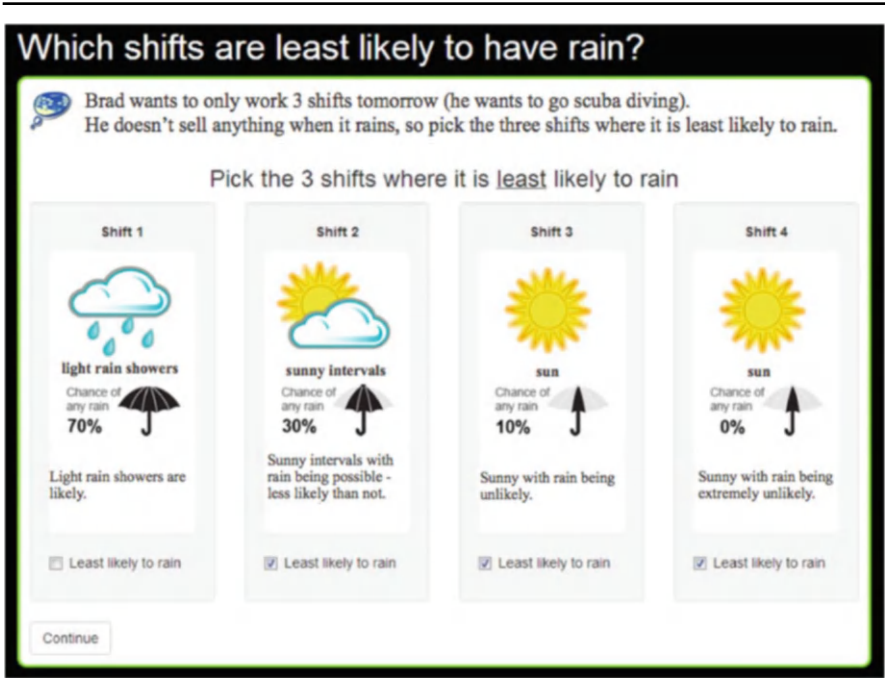


Fig. 4.6 Simple weather forecast (text and graphics); courtesy of Dmitra Gkatzia

the effectiveness of presenting clinical data to clinicians via data visualisations or (human-written) textual summaries of data and found that clinicians who saw the human-written text summaries made better decisions. Van de Meulen et al. [204] presented clinical data as either (A) data visualisations, (B) computer-generated (NLG) textual summaries, or (C) human-written textual summaries and found that doctors who saw the human-written textual summaries again made the best decisions. There was little difference overall in the quality of decisions made by doctors who saw the data visualisations and doctors who saw the computer-generated NLG texts. However in some scenarios the visualisations led to better decisions, and in others the NLG texts led to better decisions. This suggests that the best strategy is to combine the two, which matches what Gkatzia et al. found.

Of course, all of these experiments were done based on specific visualisations and NLG summaries; it is possible that different results would have been seen with different visualisations and/or NLG summaries. But it is plausible that a combined text+graphics presentation of information is usually best, not least because we know that which presentation is best depends on the scenario and also on the user (e.g. verbal vs. visual thinker); hence a combined ‘multi-modal’ presentation will be more robust across scenarios and users.

4.4.2 Other Use Cases

Balloccu and Reiter [11] looked at graphical vs. text+graphics presentation of dietary information in an app which gave people feedback about their diet. They found that users had better understanding of the dietary data when it was presented using a mixture of text and graphics.

McKeown et al. [128] looked at a number of use cases, including generating maintenance instructions and generating briefs which summarise a patient’s status after an operation. They again suggest that the best strategy is to mix text and graphics.

4.4.3 Combining Text and Graphics

The above papers suggest that a multi-modal presentation of information, combining text and graphics, is usually the most effective. Of course this raises the question of how to best combine text and graphics.

The simplest approach is to generate the text and graphics independently, so that the user sees text and graphical summaries of the data; this approach was used by Gkatzia et al [70] (Fig. 4.6). It allows users to focus on whichever media (text or graphics) works best for them in their current context; also if they struggle to understand something which is presented graphically, they can look at the text, and vice versa.

It is also possible to present different kinds of information in the two media. For example, we can use the textual descriptions to describe what-if analyses, causal links, background information, and other things that are difficult to describe in graphs, while using the graphs to communicate raw numerical data (which is awkward to do in words). This approach is common in business intelligence (Sect. 7.3), and there are a number of commercial tools which essentially explain business data (such as sale and profits) using a mixture of graphs and words; sometimes the textual information is intended to expand and/or clarify the data visualisations [119]. In interactive chatbot contexts (such as Balloccu and Reiter [11]), the user may be able to explicitly request textual or graphical presentations of data. We might describe this approach as *loose coupling*; the text and graphics communicate different information, but they can be examined independently. In other words, the textual component makes sense even without the graphical component and vice versa.

A nice example of this from a research system is SaferDriver [28], which generated feedback reports for drivers about speeding and other unsafe driving behaviour, based on GPS data. As shown in Fig. 4.7, SaferDriver used NLG texts to give summary information, key insights, and encouragement/praise, but it used a map to show detailed driving data.

In the 1990s there were a number of research projects which looked at close integration of text and graphics, where the text component did not make sense without the graphical component and vice versa. Much of this work focused on instruction giving [56, 128, 206]; I summarise some of it in Chapter 7 of my 2000 book [162]. This approach is less common in recent work.

4.5 Requirements Acquisition

Since requirements are very important, a key question is how developers and system builders can understand user requirements. There is of course a large literature in the software engineering community on requirements acquisition for building software systems (for example Wiegers and Beatty [217]), most of which is applicable to building NLG systems. There is also some work explicitly on understanding requirements for NLG.

4.5.1 User Studies

A very useful technique for understanding requirements is to conduct *user studies*. This is a powerful technique which is widely used in software engineering and interface design.

Driving Report 31 August - 6 September

You drove 170 kilometres in five hours and 40 minutes during the last week. You managed to reduce the number of acceleration incidents per kilometre by **nearly 10%**, well done!



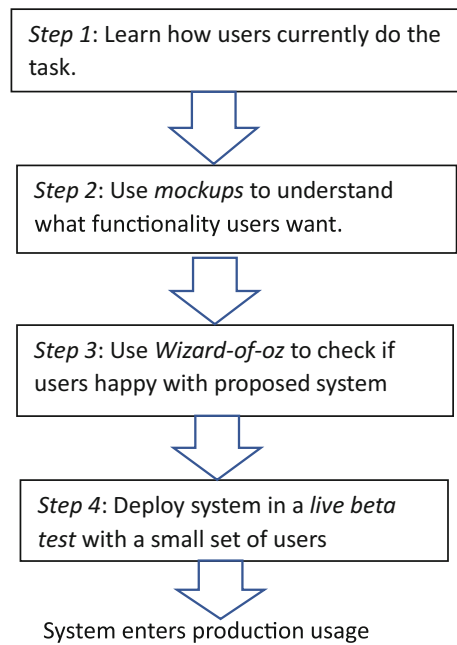
You didn't do any serious speeding, well done! However, you **speeded on 31 occasions, 15 times** on weekdays on King Street.

You **accelerated or braked harshly 103 times**, mostly **on weekdays on A123** and **on weekdays on roads with 50 km/h speed limit in Aberdeen**.

Fig. 4.7 Combining text and graphics in SaferDriver; courtesy of Daniel Braun

Knoll et al. [95] describe a set of user studies they performed in order to understand requirements for Note Generator (Sect. 1.3), a system which summarised doctor–patient consultations. They divide the process into several stages (Fig. 4.8).

Fig. 4.8 User studies process, inspired by Knoll et al. [95]



Step 1: Analysis of How Humans Currently Do the Task If an NLG system is (semi-)automating a task which is currently done manually, then the first step is understanding how humans currently perform the task. Knoll et al. conducted one-hour semi-structured interviews with seven clinicians in order to understand how they manually did the task, where they needed help, and what the difficult cases were; they also created several *personas* based on the people they interviewed.

The difficult cases (for writing up doctor–patient consultations) included cases where patients were not truthful or exaggerated symptoms, cases where patients had multiple problems, and cases where patient’s non-verbal behaviour communicated important information. In general understanding hard cases is essential in requirements analysis; developers need to ensure that they have a strategy for dealing with hard cases (which could be reverting to human authorship).

Stage 2: Mockups Once developers have an idea what an NLG system will do, the next step is to get feedback from users and stakeholders to verify that this functionality is useful and meets user/stakeholder needs. A common practice in interface design, which Knoll et al. use, is mockups; they show users simple mockups of systems and used this to get feedback on what interface and functionality is best. Showing mockups is cheap and can be done very early in the system-building process.

Amongst other things, Knoll et al. discovered that users (doctors) wanted the consultation summary to be generated and updated in real time, as the doctor and

patient spoke. In other words, doctors wanted to see a summary of the consultation as it progressed, not just at the end. This had an impact on the technology used in the final system.

Stage 3: Wizard of Oz Knoll et al. then built a *Wizard of Oz* system, where users thought they were interacting with the NLG system, but in fact the summaries were produced by a human *Wizard*, who wrote the texts which the users thought were produced by the NLG system. This is again a common technique in interface design. It is a great way to get feedback on a proposed app (does it meet user needs?) at an early stage, before the app is built.

In this case, the Wizard of Oz study showed that different users used the system in different ways. For example, some users essentially used a post-editing workflow (Sect. 4.3.2), that is they updated the computer-generated summary. Other users used a drafting workflow (Sect. 4.3.3), that is they wrote their own summary, and copied over material from the computer-generated summary where it was helpful.

Stage 4: Live Beta Test The final user-study stage was to deploy the complete NLG summarisation material in a limited test with a small number of clinicians and monitor how clinicians used it and what the problems are. This is similar to *alpha* or *beta* tests in classical software development.

Amongst other things, the beta test showed the doctors did *not* generally use the NLG system in difficult consultations of the sort described above; in such cases they reverted to manual note writing. This suggests that it is acceptable (but of course not ideal) for the system to focus on handling ‘easy’ cases well.

Production Usage Once the above stages are completed and the system has been updated based on feedback and observation, it is time to start the process of deploying the system in full production usage! Moramarco [137] describes the production deployment of Note Generator.

Of course there are other techniques which can be used in user studies, beyond those discussed in Knoll et al. For example in other projects we have used focus groups to help us understand what users wanted [191].

4.5.2 Manual Corpus Analysis

User studies focus on what users want and need and are usually the best technique for high-level requirements analysis. It can also be useful, especially when building rule-based NLG systems, to try to elicit detailed requirements by manually analysing a small set of high-quality input–output pairs. This process is called *manual corpus analysis* [162].

This process is as follows:

1. Obtain a small *corpus* of system inputs (typically 10–20 texts, ideally as varied as possible) and high-quality outputs for these inputs. These outputs are called *target texts* and are usually written by a human domain expert.

Sentences (or in some case phrases) are annotated with one of the following categories:

Static: Always present, does not change. An example is the sentence *Remember that past performance is not indicative of future results* in Figure 4.10; this is a boilerplate fixed sentence which needs to be present for legal reasons, and never changes.

Data: Directly communicates part of the input data. An example from Figure 4.10 is the sentence *The FTSE rose by 20 points yesterday*; this directly communicates basic stock market data.

Insight: Communicates insights derived from the input data. An example from Figure 4.10 is the sentence *The rise was largely driven by an increasing values of energy-related stocks*; this is the result of a key-drivers-and-offset analysis of the stock market data.

Background: Communicate background information which is available on the Internet or within a language model. An example from Figure 4.10 is the sentence *Crude oil prices are increasing rapidly*; this is not in the stock market data but is related information which can be obtained from the Internet.

Unavailable: Communicates information which is not available to the system. An example from Figure 4.10 is the sentence *Investors who bet on oil companies are very happy*; this is a plausible guess, but the system does not have access to the emotional state of millions of investors.

Fig. 4.9 Simple annotation scheme for manual corpus analysis

The FTSE rose by 20 points yesterday.	(Data)
The rise was largely driven by an increasing values of energy-related stocks.	(Insight)
Crude oil prices are increasing rapidly.	(Background)
Investors who bet on oil companies are very happy!	(Unavailable)
Remember that past performance is not indicative of future results.	(Static)

Fig. 4.10 Example of annotated human-written text using the annotation scheme of Fig. 4.9. The system’s input data is stock market share prices

-
2. Ask other people (not the domain experts who wrote the texts) to annotate the target texts to indicate where information comes from. Figure 4.9 shows a simple annotation scheme which was worked well for me on many occasions.
 3. Update the target texts to remove information and insights which cannot realistically be included in the generated text; note that this judgement partially depends on the NLG technology which will be used.
 4. Analyse the annotations of the remaining material in the target texts in order to understand what content and wording is needed and how it might be produced.
 5. Repeat above if necessary.

An example of an annotated target text is shown in Fig. 4.10.

Part of this exercise is examining Insight and Background texts and deciding whether any of these are unrealistic to produce; for example the Background text in Fig. 4.10 can only be generated if the system has access to data about oil prices (i.e. it has access to commodity prices as well as share prices). Developers also need to consider whether Unavailable texts should be dropped; for example the Unavailable text in Fig. 4.10 is essentially a bit of journalistic colour which might be acceptable in general media stories but probably should not be included in financial updates aimed at professional investors (who might indeed complain that they are not feeling very happy, because they expected a larger increase in stock prices).

Developers may wish to change target texts based on the corpus analysis. Of course, any changes should be discussed with clients and users; it is also often useful to discuss them with the domain experts who wrote the target texts if this is possible. Some changes are minor and have little impact on users, but others may significantly decrease the utility of the NLG system. When developers have agreed on a set of target texts with the client, this becomes an important resource in defining what the NLG system should do.

Once such changes have been made and the target text corpus is finalised, developers can analyse it in order to better understand the type of content that will be included in generated texts, and also how it is expressed linguistically. This analysis is very useful when building rule-based NLG systems, and can directly lead to an initial set of rules.

For neural NLG, the corpus analysis is useful as a way of understanding what information and reasoning is needed to generate content, which helps specify input sources and prompts. The language/expression analysis can identify whether text needs to be in a specific genre such as ‘weatherese’ or ‘legalese’.

4.5.2.1 Issues and Limitations

One limitation of manual corpus analysis is that it generally does *not* have good coverage of boundary or ‘edge’ cases, because there are usually hundreds or thousands of such cases. We can train a machine learning model on tens of thousands of texts, but we cannot realistically do manual corpus analysis on tens of thousands of texts! If we are building a rule-based NLG system, one approach is to identify edge cases and then explicitly ask domain experts how they should be handled.

Another limitation which is inherent in analysing human-written texts is that different human writers will write different texts, even if they are given the same input data. So a text written by author A may include different Insight statements, different Background statements, etc., compared to a text written by author B. One strategy is to analyse texts written by several experienced domain experts, in order to get good coverage of what content experts think could be included in a text.

A related issue is that some human-written texts may not be very good. It is very appealing to get the corpus texts from real-world texts written by professionals. However sometimes such texts are written by less experienced domain experts working under intense time pressure, which reduces their quality. This is certainly

true of production weather forecasts; a weather forecast written by an experienced forecaster with a plenty of time is not the same as a weather forecast which is quickly produced by a novice forecaster who is struggling to write a large number of forecasts in a short amount of time. If text quality is a concern, it may be useful to ask an experienced domain expert (e.g. forecaster) to check corpus texts and remove low-quality ones.

Last but not least, the above process requires cooperation and support from domain experts, but sometimes they are hostile to the AI/NLG initiatives because they see them as a threat to their jobs. This is a *change management* issue (Sect. 7.1.5), not a technical one, and will require sensitivity and addressing the concerns of the individuals who are supposed to be supporting the project.

4.5.3 Stakeholders

When working with people to understand requirements, it is important to talk to *stakeholders* as well as direct users. For example, a system which summarises doctor–patient consultations (Sect. 4.5.1) must be acceptable to the doctors who will use it! But it must also be acceptable to managers who look at the ‘big picture’ impact of the system, and to legal and risk management specialists who assess the likelihood and damage (reputational as well as financial) if anything goes wrong. Many medical systems also need to be acceptable to regulators and patients.

It is important when analysing requirements to work with all types of stakeholders if possible. For example, a doctor may not care how much an NLG system costs, but a manager certainly will! Likewise doctors and managers may not be able to estimate risks of successful lawsuits if something goes wrong, and specialists need to be consulted about this.

Personal Note

I have seen a number of NLG projects where success has been limited because not all stakeholders were consulted. The most common problem has been failure to consult users; managers think a certain type of NLG system will be useful and buy it, without considering whether end users actually want this functionality. I have also seen failures where users were consulted but management was not; we ended up building systems which users loved, but managers decided were not cost-effective.

4.6 Further Reading

There are many textbooks and other sources about software requirements in the general software engineering literature (e.g. [217]). Indeed, getting requirements wrong is a leading cause of many IT disasters, where an expensive IT system turns out to be useless in practice. A good example is the £10B NHS Connecting for Health programme, probably the biggest IT failure in the UK public sector (https://en.wikipedia.org/wiki/NHS_Connecting_for_Health).

Strickland's excellent retrospective [190] on the failure of IBM Watson in healthcare identifies many problems which essentially are due to poor understanding of what real-world users wanted the system to do.

I am not aware of textbooks or research papers specifically on requirements for natural language generation, but application-oriented books and papers which target specific NLG use cases often mention requirements. An excellent example which includes generalisable insights is Diakopoulus's book [48] on automatic journalism.

A lot has been written in the general AI literature about human–AI interaction and workflows; one recent survey is Mosqueira-Rey et al. [139]. Amershi et al. [5] give guidelines from HCI (human–computer interaction) perspective on human–AI interaction. O'Brien [142] discusses post-editing in machine translation. Diakopolus et al. [49] survey workflows for using language models in journalism. Unfortunately I am not aware of papers specifically on post-editing and other workflows in NLG, other than the ones cited above in Sect. 4.3. There are numerous blogs and white papers about human–AI interaction from commercial AI companies; as always with commercial material, these are often very well written but may not be completely objective.

An increasing number of papers are being published which explore different quality criteria for NLG, but I recommend that readers interested in this topic start with Howcroft et al. [81] and Belz et al. [16]. Gehrmann et al. [68] is primarily a survey of NLG evaluation, but also has some good insights about quality criteria.

My 2000 book [162] includes a fair amount of material on text and graphics. Much of this is out-of-date, but I think the discussion of basic issues and concepts is still relevant today. In 2024, the best work on combining NLG and information graphics is often done commercially, by companies (such as Arria, my company) which develop solutions which combine these in real use cases. One example of a system which Arria helped to develop is described in Sect. 7.3.1 of this book.

For requirements acquisition, Knoll et al. [95] is the only paper I am aware of which explicitly examines requirements acquisition processes for NLG. There are of course many general textbooks about software requirements (as above), which discuss requirements acquisition. Textbooks about interaction and UI design in HCI (e.g. Rogers et al. [172]) can also be useful.

Chapter 5

Evaluation



A key issue in NLG is *evaluation*, in other words assessing how well an NLG system works and meets its requirements. Does it produce texts that are acceptable in its use case, based on the relevant quality criteria (Sect. 4.1)? Evaluations can also assess whether a system is suitable for its expected workflow (Sect. 4.3) and if it is acceptable to different stakeholders (Sect. 4.5.3).

In addition to assessing the effectiveness of an NLG solution, evaluation can also highlight where systems are weak and need to be improved. For example an evaluation of an NLG system may show that its texts have good readability but poor accuracy; this is very useful information for developers who are trying to improve the system.

In this chapter I first give an example of an evaluation (Sect. 5.1) and then look at fundamental evaluation issues and concepts (Sect. 5.2). I then discuss evaluation with human subjects (Sect. 5.3), evaluation based on automatic metrics or algorithms (Sect. 5.4), and evaluation based on real-world impact (Sect. 5.5). I also look at some topics that are important in many commercial evaluations, but less important in many academic evaluations, such as system cost (Sect. 5.6). I conclude with some general advice on doing good evaluations (Sect. 5.7) and suggestions for further reading. Throughout the chapter I often refer to research on evaluating machine translation systems, which is well developed and provides many insights that are relevant to evaluation of NLG.

Evaluation is related to *software testing* of NLG systems (Sect. 6.2). However, the focus of evaluation is on trying to quantify the performance of an NLG system (usually on a set of relevant quality criteria), while the focus of software testing is on identifying specific cases and scenarios where an NLG system behaves unacceptably.

Dear Ms Cameron

Thank you for taking the trouble to return the smoking questionnaire that we sent you. It appears from your answers that although you're not planning to stop smoking in the near future, you would like to stop if it was easy. You think it would be difficult to stop because *smoking helps you cope with stress, it is something to do when you are bored, and smoking stops you putting on weight*. However, you have reasons to be confident of success if you did try to stop, and there are ways of coping with the difficulties.

Fig. 5.2 First paragraph of an example STOP leaflet [168]

variety of smokers, including those who did not have Internet access, the experiment was paper-based. Subjects filled out the questionnaires on paper and posted them; researchers scanned the questionnaires, ran the software, printed the leaflets, and posted them back to the subjects.

To evaluate STOP [110], the researchers recruited 2553 subjects in the Aberdeen area who smoked, asked them to fill out the questionnaire, and then sent each of them either (A) a STOP leaflet, (B) a default leaflet that was not personalised based on their questionnaire, or (C) a simple 'thank you for being in our study' letter. The reason for the different responses was to check if people who got STOP leaflets were more likely to stop smoking than people who got something else; these were *control groups* or *baselines*. Of course some people are going to stop smoking regardless of the NLG system, so it is not very useful to just measure how many STOP recipients managed to stop smoking. It is much better to be able to compare smoking-cessation rates in STOP recipients against smoking-cessation rates in people who did not get STOP leaflets.

Anyways, 6 months after sending out the leaflets and thank-you letters, smokers were contacted and asked if they had stopped smoking. If they said they had stopped, they were asked to provide a saliva sample, which was tested for nicotine residues (this is necessary because sometimes people are not truthful about smoking cessation). Subjects who did not respond or who did not provide saliva samples were assumed to be still smoking.

This gave data on how many people in each of the groups (STOP leaflet, default leaflet, and thank-you letter) had managed to stop smoking:

- 3.5% of people who got STOP leaflets managed to stop smoking
- 4.4% of people who got default leaflets managed to stop smoking
- 2.6% of people who just got thank-you letters managed to stop smoking

In other words, the STOP leaflets seemed to be *less* effective than the non-personalised default leaflets. This difference was not *statistically significant* (Sect. 5.2.3), but still the evaluation provided no evidence that the STOP leaflets were worthwhile; it was a *negative result*. Negative results are disappointing, but it is important that they are published, not least because they warn other researchers that some research directions may not be productive.

Personal Note:

We published STOP's negative findings in both a medical journal [110] and an AI journal [165]. Medical journals routinely published negative results, but they were almost unheard of in the AI literature at the time (early 2000s); indeed I had several discussions with the editor about how to best present a negative result to the AI community. Thankfully it is more common (although still unusual) for negative AI results to be published in 2024.

5.2 Fundamentals

5.2.1 Stakeholder Perspective

Developers must keep in mind that different stakeholders care about different things (Sect. 4.5.3). This means that an evaluation that is acceptable to one stakeholder may not be useful to a different stakeholder.

For example the Babytalk project (Sect. 2.2.2) was a collaboration between computer scientists, doctors, psychologists, and a software company (not Arria)¹ to build NLG systems which summarised data from a baby's electronic patient record. Different collaborators cared about different things from an evaluation perspective [161]:

- *Doctors and medical researchers* wanted to know if Babytalk was medically effective. They wanted to evaluate whether using the system led to better patient outcomes (or reduced clinical workload).
- *Psychologists* were interested in the effectiveness of textual versus graphical presentation of information (Sect. 4.4). They wanted to evaluate how presentation medium impacted decision quality.
- *Software house* wanted to know if a profitable product could be created using Babytalk technology. They wanted to assess costs, (commercial) benefits, and risks (Sect. 5.6).
- *Computer scientists* wanted to understand effectiveness of algorithms and models. Whereas other stakeholders just cared about the system as a whole, CS researchers wanted to understand how well components worked.
- *Parents* (who use the Babytalk system for families) wanted reports which were accurate and easy to read and which told them how well their babies were doing.

It was impossible to use a single evaluation to assess all of the above stakeholder concerns. The Babytalk team therefore did a number of evaluations (not all of which were published), including [83, 120, 136, 148], in order to assess different aspects of the Babytalk systems.

¹ Arria was founded several years after the Babytalk project started.

While it remains the norm in academic research to do one evaluation of a system, multiple evaluations may make sense if the system has different types of stakeholders.

Understanding what different stakeholders care about is part of requirements analysis (Chap. 4), and as such this issue should be explored *before* an evaluation is designed, not after it is carried out.

5.2.2 Hypothesis Testing

Evaluation, at least in the academic world, is usually a type of *scientific hypothesis testing*. Hypothesis testing is an integral part of the scientific method. Essentially scientists propose a hypothesis, such as ‘People who smoke are more likely to get lung cancer’ (medicine), ‘Nothing can travel faster than the speed of light’ (physics), or ‘the moon is made of green cheese’ (space science). Scientists then create experiments to test hypotheses; for example conduct a spectrographic analysis of the moon (or indeed ask astronauts to collect samples), or monitor a large cohort of people and compute statistically whether smokers are more likely to get lung cancer. In doing so, scientists discover that some hypotheses seem to be true (e.g. smokers are more likely to get lung cancer), while others are clearly false (e.g. the moon is not made of green cheese).

The type of hypothesis being tested of course depends on the field (e.g. physics vs medicine), as does the way hypotheses are tested (e.g. collecting samples vs statistical cohort analysis). However, the notion of experimentally testing hypotheses is fundamental to science and indeed is what has made the modern scientific method so successful.

From this perspective, the STOP evaluation discussed in Sect. 5.1 tested the hypothesis that ‘smokers who receive STOP leaflets are more likely to stop smoking than smokers who receive a default letter’. The evaluation showed that this hypothesis was probably not true.

In NLG, most hypotheses are based on comparing the output quality between texts produced by different systems. A typical hypothesis states that texts produced by system A are better than text produced by system B, according to quality criteria X, Y, and Z. For example if accuracy is an important quality criteria in the target NLG use case, researchers can hypothesise that texts produced by system A contain fewer incorrect statements (hallucinations) than texts produced by system B. In the STOP example, the main hypothesis was that STOP texts were better according to the *utility* quality criteria (Sect. 4.1.4).

There are other types of hypotheses which do not involve quality criteria, such as hypotheses based on cognitive plausibility (e.g. hypothesising that an NLG system will make similar mistakes to a human). These are important for some types of research, but I will not discuss these here.

5.2.3 Statistical Hypothesis Testing

Most NLG hypotheses are *statistical*. For instance, using an example from machine translation, let us say we hypothesise that Google Translate produces better English translations of Chinese news articles than Bing Translate, as judged by expert translators on the basis of accuracy and fluency. We do not expect that Google's translation will be better than Bing's on every possible Chinese news article. Instead, we usually hypothesise that when the translators compare Google's translation of an article to Bing's translation, in most cases they will prefer Google's translation.

But small difference may just be due to random chance. For example suppose we translate 1000 random Chinese news articles into English using Google and Bing, and our judges say that the Google translation is better in 510 cases and the Bing translation is better in 490 cases. Is this difference meaningful, or could it just be random chance?

In science, we answer such questions using *statistical hypothesis testing* [105]. In order to do this, we first frame a *null hypothesis*, for example that on average there is no difference between the quality of Google and Bing translations. Then we use statistical tests to measure how likely the observed outcome (Google texts are better in 510 out of 1000 cases) is if the null hypothesis is true (there is no difference on average between Google and Bing). In this case, a *binomial test* tells us that if Google and Bing texts have the same quality on average, then:

- There is a 2% chance that Google will be better on *exactly* 510 cases.
- There is a 27% chance that Google will be better on *at least* 510 cases. This is called the *one-tailed p value*.
- There is a 54% chance that either Google will be better than Bing on at least 510 cases or Bing will be better than Google on at least 510 cases. This is called the *two-tailed p value*.

In NLP, we normally consider that experimental results support a hypothesis if the two-tailed p-value is less than 5%. This is not true in the above example, so we could not conclude that Google was better than Bing if its translation was better in 510 cases (two-tailed p-value is 54%).

However, if our experiment showed that the Google translation was better in 540 cases (and Bing's translation was better in 460 cases), then the two-tailed p-value for the binomial test would be 1.2%. Since this is less than 5%, we could conclude from this experimental result that Google was probably better than Bing at translating news articles from Chinese to English.

In the STOP example (Sect. 5.1), a *chi-square test* was used to assess the difference between the smoking-cessation rate in the STOP group (30 smokers quit out of 857 total; 3.5% quit rate) and in the default leaflet group (37 smokers quit out of 846 total; 4.4% quit rate). This gave a two-tailed p-value of 35%. Since this is much higher than 5%, this means that the experiment is inconclusive. It is possible that the default leaflets are more effective than STOP, but it is also possible that they are equally effective. Indeed, it is even possible that the STOP leaflets are in fact

better, but this was obscured by bad luck; perhaps smokers who happened to get STOP letters were committed smokers who were less likely to quit than smokers who got the default leaflets.

Dror et al. [51] discuss common statistical tests in NLP research. Note that the 5% threshold is just a convention, and different areas of science use different thresholds. Psychology and medicine also generally use a 5% threshold, but particle physics uses a threshold of 0.00003%.

5.2.4 *Experimental Design, Execution, Reporting, and Follow-Up*

When we do an experiment (including evaluating an NLG system), we need to ensure that it is well designed, executed, reported, and followed up.

5.2.4.1 **Experimental Design**

Experiments must be well designed. For example if we are comparing Google Translate and Bing Translate as described above, then the experimental design would include:

- *Research questions:* What research questions and hypotheses is the experiment intended to address? For example as mentioned above, we could hypothesise that expert translators prefer Google over Bing when translating Chinese news articles to English.
- *Study type:* What type of study will we do? For example the above experiment is a human evaluation based on rankings (Sect. 5.3.1.1).
- *Subjects:* If our experiment uses people, how are they chosen, and what criteria must they satisfy? For example we may decide to use expert translators recruited from a freelancing platform such as Upwork.
- *Material:* What scenarios are evaluated; in other words, what are the 1000 texts being translated, and how are they chosen? For example we may decide to take 1000 random articles from the Xinhua news agency.
- *Procedure:* What do the subjects do? For example what user interface do they use to express their preference?
- *Analysis:* How are results analysed? For example we can use a binomial test (as described above) to compute statistical significance.

Of course a detailed experimental design would include much more information than above. We will discuss experimental designs for specific types of evaluations in more detail in Sects. 5.3.2 and 5.4.2.

A final point is that it is good practice in many fields of science to *pre-register* experiments, that is to enter experimental design details in a pre-registration website

before carrying out the experiment. This is common practice in medicine and psychology but is less common in AI and NLG [134].

5.2.4.2 Experimental Execution

Experiments must also be carefully executed. Unfortunately, many NLG experiments are not well executed because of problems such as the following [195]:

- *Code bugs*: Experiments can be invalid because of bugs in the code used to run the experiment. For example a subject is shown an input text X and ‘translations’ from Google and Bing, but because of a code bug the Bing translation is of a different input text Y.
- *Reporting errors*: Sometimes the data reported in the paper is different from the actual experimental results. For example the paper states that the Google text was preferred in 540 cases, but the experimental data shows that it was preferred in 510 cases.
- *User interface issues*: Experiments with human subjects can produce misleading results if a confusing user interface is used. For example if subjects are asked to use radio buttons to show whether they preferred the Google text or the Bing text, but the buttons are not clearly labelled, so it is not clear which button indicates a preference for the Google text.
- *Analysis bugs*: Statistical analyses can also be misleading, for example if 1500 subjects did the experiment, but the analysis only looked at the results from 1000 subjects and ignored the remaining 500.

Thomson et al. [195] give concrete examples of the above issues in published NLP research papers.

5.2.4.3 Experimental Follow-Up

Even after an experiment has been written up and published, researchers must respond to questions and otherwise support their research. This includes:

- *Respond to questions*: If readers are interested in an experimental evaluation, they are likely to have questions and want further details; the experimenters should respond to such requests. Unfortunately, many researchers do not do this. The ReproHum project [18] contacted 116 authors of academic NLP papers (all published in good venues) and asked the authors for additional information about their project. Only 45 (39%) of the authors responded in any fashion, and only 15 (13%) provided the requested information.
- *Correct problems*: Sometimes problems in experiments are discovered after results are published or otherwise released. If this happens, the researchers should publicly acknowledge the problems and fix their papers. Unfortunately, most NLP researchers do not do this. For example in the 10-year period, 2013–2022, the TACL (*Transactions of the ACL*) journal only had *one* formal correction to

experimental results and findings [195]. I personally know of more TACL papers in this period which had mistakes and should have been corrected.

5.2.5 *Research Questions*

Evaluations are intended to test research questions and hypotheses, which in NLG often take the form that system X is better than system Y under quality criteria Z; for example Google Translate is better than Bing Translate under the criteria of accuracy of their generated translations. The research question must be important scientifically (for basic research) or to stakeholders (for applications); there is no point in testing a research question which is of no interest to anyone.

From this perspective, it is essential to evaluate quality criteria that people care about; in applied NLG this means criteria that are important to stakeholders (Sect. 5.2.1). Sometimes the things that stakeholders care about are difficult to measure. For example in medicine doctors usually want to measure impact on clinical outcomes (does a system help patients), but measuring this experimentally can be a lot of work (Sect. 5.1). But regardless, if this is what people care about, it should be evaluated. There is a temptation for researchers to evaluate quality criteria that are relatively easy to measure, such as fluency (Sect. 4.1.1), but this is of limited value if it is not what stakeholders care about.

Another point is that if the goal is to demonstrate that a new NLG system is better than existing NLG systems (sometimes called *baselines*), it is essential to compare the new system to the best (state-of-the-art) existing NLG system(s); comparing the new system to obsolete systems is not valuable or interesting. For example if the goal is to show that a new language model can generate better texts in some contexts than existing models, then the new model should be compared to the best existing models, not to obsolete models such as GPT2.

Personal Note:

The above points may seem obvious, but I see many research papers that evaluate unimportant but easy-to-measure quality criteria and/or compare a new system against obsolete baselines. Such experiments are easy to run and produce impressive results, but they are not useful either scientifically or practically.

5.2.6 *Replication*

Scientific experiments need to be *replicable*. In other words, other researchers should be able to repeat an experiment and get similar results. This is fundamental to

the scientific method; experiments that cannot be replicated have limited scientific utility and validity.

Much has been written about replicability in different fields of science [219]. In NLG, replication requires very detailed information about the system and experiment, including how data is preprocessed, the exact libraries and options used for automatic evaluation, the exact user interface used for human evaluation, etc. Seemingly small differences can lead to big differences in outcomes [15].

Various data sheets have been proposed to help gather this information [67, 180]. Unfortunately, at least in my experience it is very rare for all the necessary details to be provided in papers and associated material such as data sheets. Usually researchers who want to repeat an experiment need to contact the original authors in order to get all the details.

It is important to keep in mind that *exact* replication of experimental results is unlikely because of experimental noise. Certainly replicated experiments with human subjects rarely give exactly the same results, even if they use the same subjects, because people do not respond 100% consistently. For example if a subject is hungry and tired, she may give lower ratings to texts than if she is well fed and rested. Another source of noise is random seeds; any system whose output is influenced by a random seed (which includes most neural NLG systems) may generate different outputs on different runs.

So we expect to see some difference in the exact numbers if an experiment is replicated. However, replications should not lead to different statistically significant outcomes at the hypothesis level. For example if the original experiment claimed that system A produced more accurate texts than system B, with the difference being statistically significant (Sect. 5.2.3), then it would be very worrying if a replication found that system A produced *less* accurate texts than system B, with the difference being statistically significant. Such a finding would strongly suggest that there are serious flaws in how the experiment was designed, executed, or reported.

Unfortunately, some published NLG evaluations, even in good venues, do not seem to be replicable. For example as part of a shared task on replication within ReproHum [17], several labs tried to reproduce published human evaluations. The worst result was for a paper which asked crowdworkers to count the number of content errors in a text [152]; this paper re-used an evaluation methodology from an earlier paper [221]. Table 5.1 shows one result (number of errors in a four-sentence extract from a corpus text) from the original paper, the reproduction studies, and a study which measured the same thing (number of errors) using a different and better methodology. The large variation in results (the highest number is 25 times larger than the lowest) raises serious concerns about the validity of the experiment and suggests that we should not evaluate systems by asking crowdworkers to count content errors.

Table 5.1 Mean number of content errors in four-sentence extracts from corpus (human-written) texts, as measured by asking crowdworkers to count errors. Result from the original paper and two replications (one of which repeated the study using in-house subjects instead of crowdworkers). Thomson et al. [197] (Figure 5.5) measured the errors using a more rigorous annotation methodology; they reported 1.58 errors on average in a human corpus text, which is approximately 0.5 errors per four-sentence extract

Paper	Note	Mean number errors
Original paper [152]	Original	0.07
Gonzalez-Corbelle et al. [74]	Replication	0.66
Watson and Gkatzia [210]	Replication	1.52
Watson and Gkatzia [210]	Replication with different subjects	0.06
Thomson et al. [197]	More rigorous measurement	0.50

Personal Note:
One of the frustrations I have with the paper being replicated [152] is its usage of an old evaluation methodology [221], when newer and better evaluation techniques were available [194]. I have seen many other cases where papers use the latest technology to *build* an NLG system, but outdated techniques to *evaluate* the system. Sometimes this happens because the authors want to enter their system into public ‘leaderboards’ which show the best performing system at a specific task as assessed by a specific evaluation technique. Leaderboards that are based on obsolete evaluation techniques are not helpful to scientific progress.

5.2.7 Ecological Validity: Artificial Versus Real-World Context

We can do an evaluation in either artificial or real contexts; this is sometimes called *ecological validity*. For example when comparing translated texts, we can either get translators to assess them in a generic way or can ask people who genuinely need the information to use the translations for real and then assess their effectiveness.

For instance, as mentioned in Sect. 5.2.1, different stakeholders in Babytalk had different perspectives on evaluation. The psychologists wanted to do evaluations in controlled artificial settings, because this reduced the number of confounding factors when testing the impact of different media on decision-making; they believed this was more important than evaluating the system in real-world usage. However, the doctors believed that evaluating systems in real clinical usage was essential, because this gave a much better understanding of utility and potential impact; they did not trust evaluations in artificial settings. The project ended up doing both types of evaluations [83, 148].

Ecological validity is especially important if the goal is to assess real-world impact (Sect. 5.5). The real world is a messy place, and we cannot assess real-world impact if we ignore this messiness.

Personal Note:

Evaluations in real-world contexts are still rare in NLG at the time of writing, which is a shame; I hope they become more popular.

5.2.8 *Test Data: Representative, Different from Training Data*

When evaluating an NLG system, researchers need to decide which test data (scenarios) the system will be tested on; from an experimental design perspective (Sect. 5.2.4.1), this is part of choosing material. Test data should be real data or in some cases a mix of real and synthetic data; it should not just be synthetic data (Sect. 3.3.4). It should also be representative of real usage.

For example a system that summarises doctor-patient consultations should be tested in a wide variety of such consultations; it should not be tested on pharmacist-patient consultations. Obtaining representative test data can be challenging, especially in use cases (such as medicine) with strong data protection constraints, but it is essential for meaningful evaluation.

For NLG systems built using machine learning, it is also essential that they are not tested on their training data; this is a fundamental principle of machine learning (systems that are tested on training data can get perfect scores on evaluations simply by memorising their training data [177]). For example if we are evaluating texts produced by Google and Bing Translate, we should ensure that these texts are not part of the training data used to build these systems. This is called *data contamination* [173].

Unfortunately, the emergence of large language models that are trained on the Internet (Sect. 3.2.5) has made it much harder to guarantee that test data was not present in the system's training data. Prompted language models can also in some case memorise or learn from test data that is provided in prompts as examples (Sect. 3.3.5) or when models are used to evaluate texts (Sect. 5.4.3.3) [13].

I have seen many evaluations of LLMs where it seemed very likely that the test data was part of the model's training data or prompts and also many evaluations where this issue was unclear. One problem is that commercial LLMs generally do not reveal their training data and also are constantly being updated. Open-source LLMs are better from this perspective and usually clearly state what data they were trained on. For this reason, it is usually easier to evaluate open-source LLMs than proprietary commercial ones.

5.3 Human Evaluation

In general, NLG systems can be evaluated by people (human evaluation), by algorithms (metrics), or by assessing real-world impact. If an impact evaluation is not possible, then a careful and well-designed human evaluation is usually the best way to meaningfully evaluate an NLG system. However a good metric evaluation is better than a poorly designed or executed human evaluation.

In this section we look at different types of human evaluation, discuss how such evaluations should be designed, executed, and evaluated, and give some examples.

5.3.1 *Types of Human Evaluation*

There are many types of human evaluations, with more being introduced every year. But in rough terms, we can distinguish between evaluations based on (A) subjective ratings or rankings, (B) error annotation, and (C) task performance (such as decision-making).

5.3.1.1 Human Evaluation Based on Ratings or Rankings

The most common form of human evaluation in NLP asks human subjects to rate texts and/or rank a set of texts, based on one or more quality criteria (Sect. 4.1). This evaluation is based on the subject's subjective opinion.

The conceptually simplest approach is to give subjects two or more texts and ask them to rank them according to the chosen quality criteria. Figure 5.3 shows an example of this, which was used in the evaluation of the SumTime weather forecast generator [167]. SumTime generated marine weather forecasts for workers in the offshore oil industry, and this evaluation asked users to rank two texts that described the predicted wind speed (at 10 meters altitude). The numeric wind prediction (produced by a supercomputer running atmosphere simulations) was given in a table; this is in the input to the NLG system. In this example, text (a) was written by a human forecaster; indeed it was extracted from an actual weather forecast for the offshore oil industry. Text (b) was produced by the SumTime NLG system. The user was asked to say which text was preferred based on three different quality criteria (Sect. 4.1): easiest to read, most accurate, and most appropriate; they could also add comments about the texts.

Many evaluations of this type have been reported in the literature. Most follow the above structure:

- The user is given the input to the NLG system (if appropriate).
- The user is given two or more texts produced from the input (sometimes including a human-written text, as was done in SumTime); the user is not told the source of the texts.

Numerical prediction data

Time	Wind Direction	Wind Speed
0600	S	18
0900	S	19
1200	S	22
1500	SSE	23
1800	S	24
2100	S	22
0000	SSW	20

Text (a)

Wind(10M): S'LY 15-20 BECOMING 22-28 BY THIS EVENING.
LATER VEERING S-SW 18-22

Text (b)

Wind(10M): S 16-21 BACKING SSE 21-26 BY MID AFTERNOON,
THEN VEERING S BY EARLY EVENING AND SSW 18-23
BY MIDNIGHT

Please cross (or tick) one box for each questions

	(a)	(b)	both same
Which text is easiest to read?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Which text is most accurate?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Which text is most appropriate?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Comments (if any)

For clarity, less text is better. The shift of direction from S to SSE would have minimal operational impact and would probably go un-noticed unless frequent observations were being made.

Fig. 5.3 Example of ranking experiment in SumTime

- The user is asked to rank texts on one or more quality criteria. Sometimes the ‘same’ or ‘no difference’ option is given; sometimes the user is forced to make a choice.
- Sometimes the user can optionally write free-text comments about the texts.

It is good practice to randomise the order of the texts. For example if the user is shown one human text and one NLG text, then choice (a) should sometimes be the human text and sometimes be the NLG text.

Of course different user interfaces can be used; for example users can be asked to drag texts into a ranked order, instead of using checkboxes. The SumTime experiment, incidentally, was done on paper, because (at the time it was run) subjects on oil rigs and supply boats did not always have easy access to computers or the Internet.

(which was online) asked users to rate the quality of the text on two quality criteria (clarity/readability and accuracy/appropriateness), and it did not ask users to explicitly compare texts. Texts produced by different systems are rated separately.

Likert scales² (usually 5 or 7 points) are often used for rating texts. Another option is *magnitude estimation*, where users are asked to position a slider on a best-worst scale, instead of choosing a discrete option.

5.3.1.2 Evaluations Based on Annotations

Human evaluations can also be done by asking subjects to *annotate* errors and other problems in a text. Annotators often have some domain expertise and may be asked to include error type and/or severity in their annotations. Once a text has been annotated, an overall score can be computed based on the number of errors, possibly weighted by type and severity. The error distribution across types and severity, and indeed the individual error annotations, can provide useful guidance for developers who wish to improve a system.

An example is shown in Fig. 5.5. This was part of the training material given to subjects who were asked to evaluate the factual accuracy of sports stories generated by neural NLG systems [197]. Annotators were asked to find inaccurate statements, classify them (incorrect number, incorrect named entity, incorrect word, context error, not checkable, and other), and also (if possible) correct the statement.

At the time of writing, the most popular annotation scheme is MQM [59], which is used to annotate machine translation outputs. It is a more complex annotation scheme than the one in Fig. 5.5, which annotates linguistic and content errors and also assigns a severity to errors.

Annotation schemes can be domain-dependent. For example Morarmarco et al. [138] use an annotation scheme as one of several evaluation techniques for Note Generator (Sect. 1.3), which generates summaries of doctor-patient consultations. The annotation scheme includes domain-specific categories such as ‘use of not universally recognised [medical] acronyms’. Similarly Magesh et al. [118] describe an annotation scheme for the output for AI-driven legal research tools, which includes groundedness (claims are supported by citations to relevant legal documents) and correctness.

Some researchers use annotations to evaluate human-written texts as well as computer-generated texts [59, 138, 197]. This gives information about the types of mistakes made by human authors in a domain; this number is often higher than initially expected.

At the time of writing, this is a relatively new form of human evaluation in NLG, so specific annotation schemes are evolving, as are user interfaces for annotation. Some schemes allow document-level annotations, especially for missing

² https://en.wikipedia.org/wiki/Likert_scale

The Memphis Grizzlies (5-2) defeated the Phoenix Suns (3 - 2) Monday 102-91 at the Talking Stick Resort Arena in Phoenix. The Grizzlies had a strong first half where they out-scored the Suns 59-42. Marc Gasol scored 18 points, leading the Grizzlies. Isaiah Thomas added 15 points, he is averaging 19 points on the season so far. The Suns' next game will be on the road against the Boston Celtics on Friday.

List of errors:

- 2: incorrect number, should be 0.
- Monday: incorrect named entity, should be Wednesday.
- Talking Stick Resort Arena: incorrect named entity, should be US Airways Center.
- strong: incorrect word, the Grizzlies did not do well in the first half.
- out-scored: incorrect word, the Suns had a higher score in first half.
- 59: incorrect number, should be 46.
- 42: incorrect number, should be 52 .
- leading: incorrect word, Marc Gasol did not lead the Grizzlies, Mike Conley did with 24 points.
- Isaiah Thomas added: context error, Thomas played for the Suns, but context here implies he played for the Grizzlies and added to their score.
- averaging 19 points in the season so far: Not checkable. Data sources report performance per season and per game, not performance at a particular point in a season.
- on the road: incorrect word, The Suns will play at home.
- Boston Celtics: incorrect named entity, the Suns will play the Sacramento Kings

Fig. 5.5 Example text with error annotations from [194]. Each annotation includes an error type and a correction. Annotators can add explanations where useful. Box score data for this game is available at <https://www.basketball-reference.com/boxscores/201411050PHO.html>

information, in other words important information which should have been included in the text but was not present.

Personal Note:

I believe that evaluation by annotation usually gives more meaningful results than evaluation by rating or ranking and hope to see more such evaluations in the future.

5.3.1.3 Evaluations Based on Task Performance

Another type of human evaluation is to ask people to use an NLG system and assess its effect on how well they perform a task. A simple example was discussed in Sect. 4.4.1 (Fig. 4.6), where the driver of an ice cream truck decided when to work based on a weather forecast.

A more complex example is the evaluation of the Babytalk BT45 system (Sect. 2.2.2) which generated texts that summarised patient record data and were

intended to help doctors make good clinical decisions. This system was evaluated by running an experiment where clinicians were asked to decide on the best intervention (action), after seeing either a Babytalk NLG text summary of relevant patient data, a human-written summary of the data, or a visualisation [148, 204].

Note that this is not an impact evaluation in the sense of Sect. 5.5, since the experiment is done in artificial context. In the BT45 experiment doctors were asked to make decisions based purely on patient record data; this is not a real-world task, since when clinicians make decisions about patients in a real hospital, they have access to many other information sources (e.g. observing the patient, talking to nurses, previous interactions they have had with the patient) in addition to patient record data. But a task-based evaluation in an artificial context nonetheless can give good insights as to how well a system works and where it needs to be improved

It is difficult to generalise about task-based evaluations; they are very different. One fairly common type of task-based evaluation in NLG is *post-edit-time* evaluation, where researchers ask human domain experts to check an NLG text and fix problems so that it can be released to real users and measure how long this checking and editing takes. Human post-editing is fairly common (Sect. 4.3.2), and the amount of time required to post-edit a text is an important real-world measure of the utility of the NLG system. However, different people have different post-editing behaviour; some just fix mistakes, whereas others rewrite texts into a different style [187]. Also post-edit time is very dependent on the user interface and workflow, as well as the texts being edited. So it is a somewhat ‘noisy’ measure of text quality. If post-edit time is used to compare two NLG systems, it is essential that the two systems be similar from a UI, workflow, and user perspective. From a practical perspective, task-based evaluations are often more expensive and time-consuming than evaluations based on ratings or annotations. However, they can give very valuable insights on how texts influence and help users.

5.3.2 *Experimental Design*

A good experimental design is critical for human evaluations, as for other evaluations. The key steps are the ones described in Sect. 5.2.4.1:

1. Choose your research hypotheses and questions.
2. Choose the overall study type.
3. Choose subjects.
4. Choose material.
5. Decide on experimental procedure.
6. Decide on analysis.

These are described in more detail below, for human evaluations. I will use an example of evaluating a system BasketballNLG which generates summaries of basketball games. Figure 5.6 shows a high-level experimental design for this evaluation; the following sections describe each step in more detail.

Research Hypothesis

- Texts produced by BasketballNLG are better than texts produced by *StateofArtSystem* under the quality criteria of accuracy and interestingness.

Study Type:

- Annotation-based evaluation of accuracy.
- Ranking-based evaluation of interestingness.

Subjects:

- 50 subjects evaluate interestingness; ten of these (after additional training and screening) annotate accuracy errors.

Material:

- 20 games (scenarios), chosen at random, for interestingness evaluation.
- 10 games, chosen at random (but different from above), for accuracy evaluation. Summaries are produced using both BasketballNLG and *StateofArtSystem*.

Procedure:

- Experiment is run on the web.
- All subjects are given detailed instructions; subjects doing error annotation must complete a training session and get a score of 80% on a screening test.
- UI is a simple form for interestingness ranking; subjects see summaries from both BasketballNLG and *StateofArtSystem* and are asked to rank them. An online version of Microsoft Word is used for annotation; subjects annotate just one summary for each game (i.e. either the BasketballNLG summary or the *StateofArtSystem* summary, not both).
- A Latin Square design is used to decide which summary (BasketballNLG or *StateofArtSystem*) each subject sees for error annotation. For interestingness, scenarios are randomly ordered, as is the order of the texts being ranked in each scenario (i.e. whether the first text is BasketballNLG or *StateofArtSystem*).
- Simple attention check questions are included halfway through the experiment.
- At the end of the experiment, subjects can (optionally) provide free-text comments on the texts and/or experiment.
- If circumstances abort an experiment, data collected so far is kept and subjects are asked to complete the experiment at a later date.

Analysis:

- Outliers are removed using Interquartile Range (IQR) algorithm.
- Accuracy: Report average error counts for the systems being compared; use a t-test to calculate statistical significance.
- Interestingness: Report preference split from the interestingness rankings; use a binomial test to calculate statistical significance.
- Do qualitative error analysis on 5 texts with poor interestingness and 5 texts with poor accuracy.
- Analyse free-text comments using thematic analysis and report key insights.

Fig. 5.6 Experimental design for human evaluation of BasketballNLG

5.3.2.1 Step 1: Choose Research Hypotheses and Questions

A good experiment cannot be designed without knowing the core hypotheses being tested, such as ‘people who get STOP letters are more likely to stop smoking than people who get non-tailored letters’. In applied NLG, research questions and hypotheses are usually based on user requirements (Chap. 4). They are often of the form that ‘texts produced by A are better than texts produced by B under quality criteria C’, for example ‘texts produced by SumTime are better than texts produced by human forecasters under the quality criterion of *appropriateness*’. Hypotheses can also be framed around task-based outcome measures, for example ‘doctors who see Babytalk summaries make better clinical decisions than doctors who see visualisations’, or indeed ‘people who get STOP letters are more likely to stop smoking than people who get non-tailored letters’.

For example for BasketballNLG, one possible set of concrete research hypotheses is that ‘texts produced by BasketballNLG are better than texts produced by *StateofArtSystem* under the quality criteria of accuracy and interestingness’; this is two hypotheses because it includes two quality criteria. *StateofArtSystem* should be replaced by the current best system for doing this task.

5.3.2.2 Step 2: Choose the Overall Study Type

The most common study types are the ones described above, but other types are also possible, such as measuring reading time. The choice is determined by the quality criteria and also by pragmatic issues such as time and cost. In rough terms:

- Ranking and rating experiments are cheapest and quickest and can be used for most quality criteria. Opinions differ as to which is better. I personally lean towards ranking in most cases, but other researchers prefer ratings, and indeed the WMT shared task for evaluating machine translation explicitly shifted from ranking to rating in 2017 [25], after experimenting with both approaches in previous years.
- Annotation experiments are better for assessing accuracy and more generally at finding errors, but they are significantly more expensive and time-consuming than experiments based on ratings and rankings. For example the 2022 WMT shared task [96] used the MQM annotation technique in a limited way to supplement ranking-based evaluation; MQM gave more meaningful assessments, but it was not feasible from a financial perspective to do all evaluation using MQM.
- Task-based evaluations are even more expensive and time-consuming than annotation evaluations, but they are the best way to measure the effect of NLG systems on users and are better at assessing utility and related quality criteria than rankings, ratings, or annotation experiments.

For example Fig. 5.6 describes a strategy for evaluating BasketBallNLG which uses a ranking-based evaluation for interestingness and an annotation evaluation for accuracy. The expense of task-based evaluation is not justified for evaluating

sports stories (which are essentially entertainment). Annotation is better than ranking/rating for accuracy but is not normally used for interestingness.

5.3.2.3 Step 3: Choose Subjects

Human evaluations require human subjects. Some experiments can be done by any fluent speaker of the language being generated. Researchers can recruit subjects for such experiments by asking friends, family members, colleagues, or students; they can also use crowdsourcing platforms such as Mechanical Turk³ or Prolific.⁴ Other experiments require subjects with specific domain knowledge (e.g. doctors) or attributes (e.g. they smoke); such subjects are usually explicitly recruited, sometimes from collaborating organisations. For example the clinicians who evaluated Babytalk (as described in Sect. 5.3.1.3) were recruited from a specific hospital (Edinburgh Royal Infirmary) which was a collaborator in the Babytalk project.

If subjects are recruited via crowdsourcing, it is important to remember that their incentive is to do tasks as quickly as possible, since this will maximise their income. There is also a danger that crowdworkers will use chatbots to do the tasks, which makes their contribution useless for a human evaluation. Perhaps in part for these reasons, some crowdworker-based evaluations are difficult to replicate; one example was discussed in Sect. 5.2.6.

Personal Note:

I have unfortunately seen a number of cases where experiments with crowd-sourced subjects were useless. For example the subjects simply ticked boxes at random without actually doing the experimental tasks, because this was the fastest way to do the task and get paid.

Sometimes the quality of crowdworker (and other) evaluations can be improved by using techniques such as the following:

- Adding special *attention check* questions, which subjects can only answer if they are doing the experiment properly. If subjects do not answer these correctly, their data is discarded.
- Checking for ‘outliers’, such as subjects doing an experiment incredibly quickly. If this is detected, again data can be discarded.
- Recruiting a pool of trusted subjects who will be repeatedly used. For example for the annotation experiments reported in [197], Thomson et al. recruited seven

³ <https://www.mturk.com/>

⁴ <https://www.prolific.com/>

annotators on Mechanical Turk and asked these annotators to do multiple tasks over a 2-year period. Since annotators liked the work (which was interesting and also well paid by Mechanical Turk standards), they were motivated to do a good job, in order to keep on getting more such work in the future.

A general principle that seems obvious but is sometimes ignored is that researchers should treat subjects well, respect them, answer their questions, pay them (if appropriate), etc. [181]. Besides being ethically correct, a well-treated subject is more likely to provide high-quality experimental data.

Another general point is that in some cases it is important that subjects are representative of a user group. For example an evaluation of the utility of computer-generated weather forecasts should be done with a representative sample of forecast users and not just with undergraduate students (whose use of forecasts is probably not representative). In all cases, researchers should report the demographic mix (age, gender, etc.) of their subjects when they write up their experiments. These can have an impact on what subjects do [70], and indeed it may make sense to test whether males and females (for example) evaluate texts differently.

A key question is how many subjects should be recruited. In theory, this can be answered using a *statistical power calculation*, which computes the number of subjects based on expected effect size (expected difference in measured quality criteria between systems), the statistical test used in analysis, and other such parameters. However, it is rare to see power calculations used when designing NLG experiments, partially because the necessary information (e.g. expected effect size) is often not available.

In very crude terms, ranking/rating and task-based evaluations usually have at least 20 subjects, especially if subjects do not need specialist expertise. Annotation-based evaluations, and evaluations with domain experts, can be done with fewer subjects, sometimes as few as 3 or 4. These are *minimum* numbers; it is much better to have 50 or 100 subjects in a ranking/rating evaluation.

For example Fig. 5.6 describes a strategy for evaluating BasketBallNLG which uses 50 subjects to rank basketball texts on the basis of interestingness. From this group, ten people are asked to annotate texts for accuracy; researchers look for subjects who have good knowledge of basketball and ask them to pass a screening test. The test could ask subjects to annotate a summary with known errors and only accept people whose annotation is 80% correct (precision and recall compared to the correct annotation).

5.3.2.4 Step 4: Choose Material

In most human evaluations of NLG, subjects are shown texts generated in specific *scenarios*. For example in the Babytalk evaluation doctors evaluated texts generated from patient record data about a specific baby over a specific time period; a scenario here might be *baby JOHNSMITH23 at 1600–1645 on 3 September 2004*. In the SumTime evaluation weather forecast users evaluated forecasts for specific locations

on specific days; an example scenario here would be *Brent (offshore oil field) on 1 August 2000*. Hence an important design issue in human evaluations is choosing the specific scenarios used in the experiment.

One approach is to choose scenarios randomly. For example in the SumTime context, if there is forecast data for ten locations over a 100-day period, we can randomly select location 5 on day 59, location 1 on day 80, etc. (I generated these using a random number generator).

In other cases it is important that scenarios are diverse and cover important categories and/or edge cases. One possibility in such cases is to randomly select within a category. For example in the Babytalk BT45 evaluation mentioned in Sect. 5.3.1.3, it was essential that the selected scenarios covered eight different recommended interventions (scenarios where the recommended clinical action was to increase oxygen levels, scenarios where the recommendation was that the doctors do nothing, etc.). So the researchers created sets of potential scenarios which met each of these criteria (e.g. one set was all potential scenarios where the recommended action was to increase oxygen levels) and then randomly selected experimental scenarios from each of these sets.

As with subject choice, another decision is how many scenarios to use. Sometimes this can be computed from the number of subjects, the number of judgements (ratings, rankings, or annotations) made by each subject, the number of texts in each scenario (often one per system), and the number of times each text is judged, as follows:

$$\begin{aligned}\text{NumberOfJudgements} &= \text{NumberOfSubjects} \\ &\quad * \text{NumberJudgementPerSubject} \\ \text{JudgementsPerScenario} &= \text{TextsPerScenario} * \text{JudgementsPerText} \\ \text{NumberScenarios} &= \text{NumberOfJudgements} / \text{JudgementsPerScenario}\end{aligned}$$

For example for the annotation portion of the BasketballNLG evaluation, scenarios can be chosen at random. In terms of numbers, assume:

- Five annotators (subjects)
- Eight texts annotated by each annotator
- Two texts per scenario (one produced by BasketballNLG and one produced by state-of-art baseline)
- Each text annotated by two people (to reduce the chance of missing something)

The above formula suggests ten scenarios.

$$\begin{aligned}\text{NumberOfJudgements} &= 5 * 8 = 40 \\ \text{JudgementsPerScenario} &= 2 * 2 = 4 \\ \text{NumberScenarios} &= 40 / 4 = 10\end{aligned}$$

5.3.2.5 Step 5: Decide on Experimental Procedure

Experimental procedure includes many different things, such as:

- *Delivery*: How is the experiment run? Is it done over the web, online but in an experimental room, or on paper? Web-based experiments are usually the easiest to run, but experimenters have more control and can detect problems better if they are in the same room as the subjects; they can also observe subjects, which is sometimes useful. Some quality criteria, such as reading time, are difficult to accurately measure in web-based experiments.
- *User Interface and Training*: What user interface (UI) do subjects use, and what training (if any) are they given to explain the task and UI? It is important to get this right, since it is essential that subjects understand what they are supposed to do.⁵ Piloting the experiment (Sect. 5.3.3.2) can often detect UI problems.
- *Text selection*: Which texts do subjects see, and in what order? It is often good practice to use a *Latin Square design* [1] to ensure that each subject sees a balanced number of texts from each system (since some subjects are more generous in their ratings or more skilled at the task). If the experiment involves ranking texts produced by different systems, then the order in which the texts are shown should be randomised (since some subjects may have a preference for preferring the first text that they see).
- *Attention checks*: Does the experiment include attention-check questions? These are questions with clear answers whose purpose is to ensure that subjects are taking the experiment seriously and not just clicking at random. Attention checks are common in web-based experiments using crowdworkers, where there is a fear that crowdworkers will try to do the experiment as quickly as possible in order to maximise their income (Sect. 5.3.2.3). Data from subjects who fail attention checks should be discarded.
- *Dealing with disruptions*: What should happen if something goes wrong? For example a network outage or fire alarm disrupts an experiment, or a subject has to drop out of an experiment because she is not feeling well? It is not possible to plan for every potential event, but it is worth thinking at least in general terms about how to handle such disruptions.

Figure 5.6 includes a high-level experimental design for evaluating BasketBallNLG. Of course many details need to be fleshed out, such as the exact design of UI forms.

5.3.2.6 Step 6: Decide on Analysis

Another key experiment design issue is how experimental data will be analysed; this should be decided *before* the experiment is run and data is collected. Analyses should focus on testing hypotheses (the ones chosen in Step 1). From an insight perspective, it is important to qualitatively as well as quantitatively analyse the experimental data.

⁵ <https://ehudreiter.com/2024/05/28/human-eval-subjects-must-understand-the-task/>

Most NLG hypotheses are stated in terms of system A having a higher score than system B. Score should of course be computed and reported, but it is also important to identify and remove outliers and to perform statistical significance testing.

An *outlier* is a data point that is very different from the other data points. Sometimes outliers are genuine; if so, examining them can lead to major insights about hypotheses. However, outliers often signify experimental flaws, for example a subject who did not take the experiment seriously (and just clicked things at random) or who misunderstood the task. Such outliers are not genuine data about the hypothesis and should be discarded. Sometimes it makes sense to identify subjects (as well as individual data points) who are outliers and discard all data from this subject.

It is dangerous for experimenters to identify and discard outliers on an ad hoc basis, because the experimenter may then (perhaps unconsciously) be tempted to discard ‘inconvenient’ data points that go against the experimental hypothesis. For this reason, researchers should specify an outlier identification policy *before* running the experiment, such as the widely used *IQR* method.⁶

Once outliers have been removed, researchers can compute the mean (or median) score for each system. Statistical significance (Sect. 5.2.3) of the differences in scores should also be computed.

Different statistical tests are used in different experiments [51]. Some simple advice that works for many (not all) experiments is:

- When comparing numerical scores, such as the number of errors in a text, use a *t-test* if comparing two systems and an *ANOVA* if comparing more than two systems.
- When comparing Likert-scale ratings of texts in a ratings experiment, use a *Mann Whitney test* if comparing two systems and a *Kruskal-Wallis* test if comparing more than two systems.
- When comparing preferences in a ranking experiment, use a *binomial* test if just two systems are being compared (ignore ‘no difference’ choices if these are allowed). If more than two systems are being compared, use a *Friedman* test.
- When comparing categorical outcomes (e.g. how many people stopped smoking in different groups), use a *chi-square* test (regardless of the number of systems being compared).

If in doubt about which statistical test to use, you should consult with a statistician.

Once the main hypotheses have been tested, researchers can also look for other interesting patterns in the data. This is often done using Exploratory Data Analysis (EDA) techniques [199], where a researcher explores the data and looks for patterns using a general-purpose visualisation and analysis tool such as R (people with limited programming expertise often use Microsoft Excel). When reporting results, it is important to distinguish between formal results from hypothesis testing and additional insights from EDA.

⁶ https://en.wikipedia.org/wiki/Interquartile_range

In addition to numerically analysing data, it is also important to *qualitatively* analyse data. One technique that I recommend is *qualitative error analysis*; this involves taking a few example texts (often including texts with poor scores) and qualitatively analysing them. For example if the experiment is based on annotating errors (Sect. 5.3.1.2), then the quantitative analysis will be based on counting errors (perhaps weighted by severity), while the qualitative analysis explores specific individual errors which shed light on what can go wrong. If the experiment is based on Likert ratings of readability (Sect. 5.3.1.2), the quantitative analysis is usually based on mean score, while the qualitative analysis could discuss specific individual texts which got low Likert ratings from subjects. Qualitative analysis does not prove or disprove hypotheses, but it does give insights on what the problems are in generated texts, which can help developers improve the relevant systems.

In many experiments subjects are asked to provide free-text comments as well as specific ratings, rankings, annotations, etc.; I definitely recommend doing this when possible. These free-text comments are another good source of insights into problems and areas for improvement.

If subjects provide free-text comments, these can be analysed using a thematic analysis [29]. Another approach is simply to read the comments and identify ones which provide interesting insights about the texts and hypotheses; these can be summarised in the experimental report, including quotations from especially interesting comments. When doing this, it is important to ensure that quotations are anonymous and do not reveal the authors' identity.

Figure 5.6 includes a possible analysis plan for BasketballNLG.

5.3.3 Issues in Human Evaluation

Regardless of the experimental design and type of evaluation, there are a number of generic issues that are important in human evaluation.

5.3.3.1 Inter-annotator Agreement

It is good practice in human evaluation to ask multiple subjects to assess the same texts and then measure how well they agree. That is if subjects A, B, and C evaluate a text T under the relevant quality criteria, how likely are they to agree, for example to all say that the text is excellent? This is called *inter-annotator agreement*.

Inter-annotator agreement is most commonly measured using a *kappa* score, typically Cohen's kappa for agreement between two annotators and Fleiss kappa for agreement between more than two annotators; Krippendorff's alpha can also be used. McHugh [127] gives a nice summary which is accessible to non-statisticians of kappa and related measures, how they are used to measure annotator agreement, and some issues and concerns. Most statistical software packages include functions

for calculating kappa and alpha, and it is good practice in human evaluations to report a kappa or an alpha score.

If the experimental design specifies that most texts are only evaluated by a single subject, then it may make sense to get a subset of texts annotated by multiple annotators, so that inter-annotator agreement can be reported.

The kappa statistic is a number between 0 and 1. McHugh [127] suggests interpreting kappa as follows:

- Above 0.90: *Almost perfect*
- 0.80–0.90: *Strong*
- 0.60–0.79: *Moderate*
- 0.40–0.59: *Weak*
- 0.21–0.39: *Minimal*
- 0–0.20: *None*

Many NLP researchers use a more lenient interpretation, where for example a kappa statistic of 0.4 is considered to be Moderate rather than Weak. Of course the interpretation of kappa depends on the circumstances and type of experiment, but in general I recommend that researchers use McHugh’s more conservative interpretation.

If inter-annotator agreement is weak (less than 0.60), this means that different annotators are assessing texts quite differently, which is not good; replication (Sect. 5.2.6) may be less successful if agreement is low. Of course human beings are different, which means that no two individuals will assess a text in exactly the same way, but nonetheless we have more confidence in the result of a human evaluation if there is good agreement between annotators.

Agreement can often be increased by changing the experimental design, for example adding attention checks to filter out subjects who are not taking the experiment seriously and/or providing training and clear guidance to subjects. Agreement is also often higher for simple experimental tasks than more complex tasks.

Personal Note:

In my experience, inter-annotator agreement is usually higher for annotation-based evaluations (Sect. 5.3.1.2) than for rating/ranking evaluations (Sect. 5.3.1.1), which is one reason I prefer not to use rating/ranking evaluations if I have a choice.

5.3.3.2 Piloting Experiments

Experiments need to be carefully designed and also executed, as described in Sect. 5.2.4.2. Since human experiments are expensive and time-consuming, it

usually makes sense to first conduct a small-scale *pilot* experiment and use this to investigate problems such as code bugs, randomisation errors, and confusing user interfaces. For example researchers can use a pilot to check:

- Are the correct texts shown to users [195]?
- Does the experimental user interface (UI) make sense to subjects, and do they use it correctly?
- Is all of the expected experimental data recorded?
- Does the analysis (including outlier detection) make sense?

Personal Note:

We once ran a fairly expensive (and impossible to replicate) experiment where we discovered after completion that some key data had not been recorded [220]. If we had done a pilot first, we would have realised this and fixed our software before running the experiment!

Please remember that the point of a pilot is not to gather data, but rather to debug the experiment! Sometimes pilot data can be interesting, but the primary objective is to ensure that the experimental design works and that there are no execution errors. If you detect problems in a pilot and fix them, you may want to consider doing another pilot, since it is possible that fixing one set of bugs has inadvertently introduced new bugs.

I usually do pilots with 5–10 subjects, but it is possible to do useful pilots with just 2 or 3 subjects. However experimenters should not do pilots using themselves as subjects; it is better to get other people to be subjects. This is because experimenters know what is supposed to happen in an experiment and thus (for example) may not detect confusing instructions or user interfaces.

5.3.3.3 Research Ethics

Experiments with human subjects must be *ethical*. Amongst other things:

- Experiments should not harm subjects, third parties, or researchers. For example experiments should not normally give subjects misleading medical advice or show them pornographic or otherwise offensive material (which could happen if subjects are shown random material from the Internet).
- Experiments should not lead to sensitive personal data being published or otherwise released.
- Human subjects must agree to participate in experiments; they should be not be forced or coerced (e.g. managers cannot order subordinates to take part in an experiment).

In many contexts, experiments must be formally approved by a *research ethics* committee (called *institutional review board* (IRB) in USA). Details depend on the

type of experiment, the location, whether the experimenter works for a university or a company, and who is funding the research. Researchers therefore should consult locally about research ethics procedures and rules.

Even when formal ethical approval is not legally required, it is good practice to consider if an experiment could inadvertently harm people or otherwise be unethical. Indeed, I know of companies which are not legally required to review ethical issues in experiments but have still implemented very strict review procedures, partially to minimise legal (Sect. 3.4.4) and reputational risks if an experiment harms subjects or third parties.

Experiments may be modified in order to reduce ethical concerns; this is part of the experimental design process. For example material that experimenters would like to show to subjects can be checked beforehand for offensive content, so that only non-offensive material is actually shown to subjects.

In Aberdeen University in 2024, all experiments that involve human subjects must be approved by a research ethics committee. Some types of medical-related experiments must be approved by an NHS (UK National Health Service) ethics committee; other experiments are approved by an ethics committee within the university. Approval typically takes a few weeks for straightforward experiments, which includes most rating, ranking, and annotation experiments. Approval can take longer for task-based experiments that impact a user's actions or behaviour, experiments that involve children or other vulnerable populations, experiments that require gathering sensitive personal information, and experiments where subjects are deceived.

Personal Note:

I recently discussed with a colleague ethical review procedures for an experiment I planned to do with collaborators in China. My colleague had previously done experiments in China and explained that he had not needed to do an ethical review. However, ethical review requirements in China are changing, and reviews will be required for this type of experiment in the future; fortunately we realised this in time.

5.4 Automatic Evaluation

It is also possible to evaluate texts automatically (without human involvement), by using *metrics*, that is algorithms that assess the quality of a generated text. These can either directly assess quality criteria, or they can measure how similar generated texts are to *reference texts*, which are usually high-quality human-written texts in a corpus.

Automatic evaluation is much cheaper and quicker than human evaluation, it is also usually easier to replicate. The problem is that the results of automatic evaluation are not always meaningful predictors of real-world text quality and utility; metrics should be *validated* to assess how well they agree with trusted high-quality human or impact evaluations (Sect. 5.4.4).

Sometimes researchers carry out both an automatic evaluation on a large test set and a human evaluation on a subset of the test data. The human-evaluated subset can be randomly selected; readers will have more confidence in the automatic evaluation if it broadly agrees with the human evaluation. Another strategy is to use an automatic evaluation to find cases where the NLG system seems to do poorly and get these evaluated by humans in order to get a deeper understanding of what the problems are.

A large number of evaluation metrics have been proposed in the literature, with new ones coming out every month, so the discussion in this section will focus on principles instead of specific metrics.

5.4.1 Types of Automatic Evaluation

5.4.1.1 Reference-Based Metrics

The most common type of automatic evaluation (at the time of writing) compares generated texts against human-written reference texts. The simplest such metric is *edit distance* which simply counts how many edits (changes) need to be made to a generated text in order to match the reference text; this can be computed at either word or character level. An example is shown in Fig. 5.7, using the same SumTime weather domain as Figs. 5.3 and 5.4. The BLEU (Sect. 5.4.3.1) and ROUGE [115] metrics, which were introduced in the early 2000s but are still popular, essentially apply a more sophisticated scoring formula (based on n-grams) to word-level differences between the generated and target texts. The chrF metric [147] takes a similar approach but looks at character-level differences instead of word-level differences; it may be the best of the simple string-similarity metrics [98].

Recent reference-based metrics such as BLEURT (Sect. 5.4.3.2) and BERTScore [224] are usually trained. In other words, instead of identifying differences and applying a scoring algorithm, they use a machine learning model that is trained on human evaluation results to predict the quality of a generated text from a reference text. Amongst other things, this means that these metrics can consider semantic similarity as well as string similarity.

As explained by Kocmi et al. [98], trained metrics give judgements which are more similar to human evaluations than string-based metrics. On the other hand, the results of string-similarity metrics are easier to interpret, which may be useful for developers who are trying to improve an NLG system.

Some metrics, including BLEU, can use multiple reference texts, which allows for cases (common in NLG) when there are many acceptable output texts.

Reference text:

SSW 16-20 GRADUALLY BACKING SSE THEN BECOMING VARIABLE 10 OR LESS BY MIDNIGHT

Generated text:

SSW'LY 16-20 GRADUALLY BACKING SSE'LY THEN DECREASING VARIABLE 4-8 BY LATE EVENING

Differences:

SSW~~'LY~~ 16-20 GRADUALLY BACKING SSE~~'LY~~ THEN ~~DECREASING~~ *BECOMING* VARIABLE 4-8 *10 OR LESS* BY ~~LATE-EVENING~~ *MIDNIGHT*.

Edit count:

- Two deletions of ~~'LY~~ (one token deleted, twice)
- ~~DECREASING~~ changed to *BECOMING* (one token changed)
- 4-8 changed to *10 OR LESS* (three tokens changed)
- ~~LATE-EVENING~~ changed to *MIDNIGHT* (two tokens changed)
- No tokens added

Token-level edit distance is 8 tokens deleted, changed, or modified

Character-level (Levenshtein [112]) edit distance is 27

Fig. 5.7 Example of edit distance evaluation in weather domain

Reference-based metrics of course require good-quality reference texts. Unfortunately, in some cases insufficient attention is paid to the quality of reference texts. More fundamentally, if the NLG system is capable of generating human quality (or better than human texts), then it may not make sense to evaluate a generated text by comparing it to a human-written reference text.

5.4.1.2 Referenceless Metrics

Generated texts can also be evaluated using metrics or algorithms which do not require reference texts. A simple example is the Flesch-Kincaid reading grade level [92], which assesses the readability of a text by giving a US ‘grade level’; for example if a text gets a grade level of 9, then it should be readable by the average 15 year old. The score is calculated using a linear regression on the mean number of words per sentence and the mean number of syllables per word; these coefficients were originally calculated by fitting the regression model to a data set of how well children of different ages could read different texts (as determined by educational tests).

Recently much more sophisticated models have been built, using modern machine learning techniques, which estimate different quality criteria of generated texts. Kocmi et al. [98] say that the best of these models are almost as good as the best reference-based metrics, at least for evaluating the quality of machine

translation texts. At the time of writing, there is a lot of interest and excitement in using large language models such as GPT to evaluate texts (Sect. 5.4.3.3).

5.4.1.3 Compute Speed, Resources, and Cost

Most NLG evaluations focus on text quality, but in many contexts computational speed, resources, and cost are also important; this is usually measured either as the time taken to generate a text with a given computational resource (e.g. GPU) or the cost required to generate a text using cloud-based computational resources.

From an evaluation perspective, speed evaluations can be done using timing functions which are built into Python and other programming languages. It is good practice when doing a speed evaluation to shut down other processes and even (if possible) disconnect from the Internet; this reduces the amount of ‘noise’ in the evaluation. From a replicability perspective, when describing the experiment, it is essential to give not just full hardware specs but also details about operating system, libraries used, compilers, and so forth. It is also good practice to generate each text several times and report the mean (average) time across runs.

Cost-based evaluations of NLG are relatively unusual at the time of writing, but they can be conducted in the same way, using the metering functionality built into most cloud computing services. Of course it is not possible to disconnect from the Internet if the NLG system uses cloud computing services.

5.4.2 Experimental Design

Experimental design for automatic evaluation must address the key steps described in Sect. 5.2.4.1, except that there is no need to choose subjects (since human subjects are not needed for metric evaluations).

1. Choose research hypotheses and questions.
2. Choose the overall study type.
3. ~~Choose subjects.~~
4. Choose material, i.e. the test set that metrics are calculated on.
5. Decide on experimental procedure.
6. Decide on analysis.

An example is shown in Fig. 5.8; this is for BasketballNLG, the same system that was used in the human evaluation example shown in Fig. 5.6.

5.4.2.1 Research Hypotheses

Similar to human evaluations (Sect. 5.3.2.1), most automatic evaluations test hypotheses of the form ‘Texts produced by A are better than texts produced by

Research Hypothesis

- Texts produced by BasketballNLG are better than texts produced by *StateofArtSystem* under the quality criteria of accuracy and fluency.

Study Type:

- Reference-based metric
- Metric is not trained or fine-tuned

Material:

- 100 recent games which were played no earlier than one month before the experiment.

Procedure:

- Use BLEURT metric with standard parameters and no fine-tuning, to evaluate texts produced by BasketballNLG and *StateofArtSystem* for the 100 games.

Analysis:

- Compute system-level score by taking mean of text-level BLEURT scores for texts produced by each system (BasketballNLG and *StateofArtSystem*).
- Use a paired t-test on text-level scores to compute statistical significance of differences between BasketballNLG and *StateofArtSystem*.
- Do qualitative error analysis on the 5 BasketballNLG texts that had the lowest BLEURT scores.

Fig. 5.8 Experimental design for metric evaluation of BasketballNLG

B under quality criterion C'. Some quality criteria, such as utility, are difficult to measure with metrics, but others (such as fluency) can often be assessed with metrics.

Most automatic evaluations compare a new NLG system against an existing system. If the underlying claim is that the NLG system is better than existing systems, then it is essential that the comparison system should be a high-quality state-of-the-art system.

Personal Note:

I have seen many papers that claimed impressive results by comparing a new NLG system against an obsolete and out-of-date comparison system; this is not good science.

Automatic evaluations can also be used to guide design decisions during the development process. For example if developers want to use a prompted model but are unsure which prompt is best, they can use automatic evaluations to assess how well the system works with different prompts. In this case the research hypothesis is that a specific NLG system produces better quality texts if it uses prompt XXX instead of YYY.

5.4.2.2 Study Type

There is less variety in study types with metric-based evaluations compared to human evaluations (Sect. 5.3.2.2), but some decisions still need to be made:

- Will reference-based or referenceless metrics be used (or both)? If reference-based metrics are used, what is the source of the reference texts?
- Are metrics fine-tuned so that they work better in the target domain? If so, what data will be used to fine-tune the metrics?

5.4.2.3 Material

As with human evaluations (Sect. 5.3.2.4), scenarios must be chosen to evaluate the systems; in machine learning terminology, a *test set* must be created. But while most human evaluations use a relatively small number of scenarios (typically less than 100), metric-based evaluations can use thousands or even millions of scenarios. This is one of the strengths of metric-based evaluation; it can assess performance on a much wider range of scenarios than is possible with human evaluation.

It is common practice in machine learning to create a test set by taking a large data set and extracting part of it as test data; the rest is used to train and tune the model. However this is not always a good approach when evaluating systems built on large language models because of data contamination concerns (Sect. 5.2.8). A better approach in such contexts is to create test scenarios from new data; for example a basketball-story generator can be tested on data from recent matches. This is the approach suggested in the experimental design shown in Fig. 5.8.

If researchers have access to confidential data which was never published on the Internet because of privacy or commercial secrecy, it may be possible to use this to create test data. However it will be difficult for other researchers to replicate the experiment if the test data cannot be published.

Personal Note:

Producing good test data for testing large language models is difficult. Unfortunately, I have seen many academic papers that ignore this issue and simply test models on data which they were trained on, without even acknowledging that this is a problem. This is not good science.

5.4.2.4 Experimental Procedure

A key decision is which specific metrics will be used. I cannot give specific advice here because anything I say will be out of date by the time this book is published.

Researchers must also decide on metric parameters and preprocessing techniques such as tokenization; these can have a major impact on metric scores [149].

As mentioned in Sect. 5.2.6, one dilemma is that sometimes researchers want to use similar metrics to previously published work in order to make their results more comparable to previous research, but the metrics used in this older work are less meaningful than newer metrics. For example the BLEU metric continued to be used long after much better alternatives were available, in part because it made comparison with earlier work easier. In such cases, researchers can show results both from the best available metrics and from metrics used in earlier papers.

In some cases, when asked to compare two texts, metrics will be biased and rate the first text more highly because it is first [208]. For this reason, it is useful to vary the order in which texts are presented, which is also good practice for human evaluations (Sect. 5.2.6).

5.4.2.5 Analysis

The best analysis procedure for metrics depends on the metric chosen. Researchers should check the literature and best practice guidelines for their chosen metrics when deciding on issues such as the following:

- If a metric produces scores for individual texts, then these scores are usually aggregated into a system-level score. The obvious approach is to use the mean of the text-level scores as the system score, but there are alternatives, some of which involve weighting some texts (e.g. longer texts) more highly than others. Worst-case performance should be reported if requirements say this is important (Sect. 4.2.3).
- As with human evaluations, it is essential to perform statistical significance tests and presenting raw numbers. The best statistical test depends on the metrics, and sometimes specialised tests are needed [99].
- Small difference in metric scores, even if statistically significant, may not be meaningful; this is discussed in Sect. 5.4.4.

There are many software packages that automatically run metrics on a test set and analyse the results. It is fine to use such packages, but researchers should understand how the chosen package analyses the data.

When reporting the experiment, researchers should specify which metrics were used and why they were chosen. Details should be given about the software used to compute the metric, including parameter settings (if appropriate). If the metric software came from a library, library details (including version number) should be given.

If a human evaluation was also performed, the correlation between the human and metric evaluations should be reported.

A general point is that it is very easy to compute multiple metrics on a data set; some papers report scores for more than ten different metrics when comparing systems. If this is done, then a multiple hypothesis correction should be considered.

For example if ten different metric scores are reported, then a *Bonferroni correction*⁷ can be applied by dividing the usual statistical significance threshold of 0.05 by the number of metrics used (10). A result is only considered to be statistically significant if the p-value is less than the modified threshold, which is 0.005 in this example.

Last but not least, as with human evaluation (Sect. 5.3.2.6), it can be very useful to do a (manual) qualitative error analysis on individual texts which had poor metric scores.

5.4.3 Examples of Metrics

It is impossible to provide an up-to-date list of metrics, because new metrics are being proposed every month, and any list provided in this book would be obsolete by the time the book is published. Nonetheless, I will describe a few of the more popular metrics (at the time of writing), to give readers a better understanding of how they work.

5.4.3.1 BLEU

One of the oldest and best known reference-based metrics is BLEU [145]. The core idea of BLEU is to compute n-gram precision. That is, it counts the number of *n-grams* (Sect. 3.2.2) in the generated text which also occur in the reference text and divides by the number of n-grams in the generated text. N-gram refers to word combinations: 1-gram is just words, 2-grams are pairs of words, 3-grams are word triples, etc.

For example consider the following sentences:

- *Generated*: It will be very windy on Sunday.
- *Reference*: It will be windy on the weekend.

There are seven words and hence *unigrams* (1-grams) in the generated sentence: *It, will, be, very, windy, on, Sunday*. Five of these (*It, will, be, windy, on*) appear in the reference sentence. So the 1-gram precision of the generated sentence is $5/7 = 0.71$

We can also look at *bigrams* (2-grams) in the generated sentence: There are six of these (*It will, will be, be very, very windy, windy on, on Sunday*). Three of these are in the reference text (*It will, will be, windy on*), so the 2-gram precision of the generated sentence is $3/6 = 0.50$.

The real BLEU algorithm incorporates a number of adjustments to the core n-gram precision formula, including allowing multiple reference texts to be used and specifying how a system-level score can be calculated from scores for individual texts (like the example above). There are many implementations of BLEU available

⁷ https://en.wikipedia.org/wiki/Bonferroni_correction

in NLP toolkits and repositories, and almost all researchers use one of these instead of coding BLEU from scratch. By far the most common n-gram size is 4, and published BLEU scores are based on 4-grams unless otherwise indicated.

A number of other n-gram-based metrics have been proposed in the literature, including ROUGE and chrF. Kocmi et al. [98] recommend chrF [147] as the best such metric.

5.4.3.2 BLEURT

BLEURT [179] is a more recent reference-based metric which is trained on data. In other words, while BLEU essentially uses an explicit algorithm to compute its score, BLEURT uses machine learning techniques to train a model which produces scores.

The details are again complex, but the core idea is to take the BERT language model [47] and fine-tune it for the score generation task. Fine-tuning requires a data set which contains texts and corresponding human evaluation scores.

Since there is a limited amount of data which provides high-quality human evaluations of generated texts, BLEURT first fine-tunes the model on synthetic data (Sect. 3.3.4), which it produces by perturbing Wikipedia sentences (for instance by back-translation, see example in Fig. 3.7). It then uses BLEU and other older metrics to estimate the quality of the perturbed sentences. The quality of this synthetic data set is not great, but it is very large, and tuning on it makes BLEURT more robust.

After the model has been fine-tuned on synthetic data, it is then further fine-tuned on actual (text, human ratings) pairs from one of the WMT tasks.

BLEURT can be used ‘off the shelf’, or it can be further fine-tuned for a specific NLG task, by giving it appropriate (text, human rating) training data.

5.4.3.3 Evaluation Using Large Language Models: GEMBA-MQM

At the time of writing, there is a lot of interest and excitement in using prompted large language models (such as GPT) to evaluate the quality of generated texts, by essentially just asking the model to do this in its prompt (and possibly including some examples). In some contexts this can give more meaningful results than simple human evaluations [223]. This space is evolving very quickly, so I will just give a representative example, which is GEMBA-MQM [97].

GEMBA-MQM asks the GPT4 language model to annotate texts (output of machine translation systems) using the MQM annotation scheme [59]. MQM annotation by human translators is (at the time of writing) probably the most rigorous and meaningful evaluation technique in machine translation, but it is expensive and time-consuming.

GEMBA-MQM asks GPT4 to perform an MQM annotation, using a prompt which describes the MQM protocol in a fairly straightforward manner (an extract is given in Fig. 5.9) and also gives three examples of MQM-annotated texts.

(System) You are an annotator for the quality of machine translation. Your task is to identify errors and assess the quality of the translation.

```
(user) {source_language} source:\n
      '{source_segment}'\n
      {target_language} translation:\n
      '{target_segment}'\n
      \n
```

Based on the source segment and machine translation surrounded with triple backticks, identify error types in the translation and classify them. The categories of errors are: accuracy (addition, mistranslation, omission, untranslated text), fluency (character encoding, grammar, inconsistency, punctuation, register, spelling), [etc]

Fig. 5.9 Part of the GPT4 prompt used by GEMBA-MQM, from [97]. {source_language} is replaced by the language of the text being translated, {source_segment} is replaced by the text being translated, etc. The prompt also includes three examples of MQM annotated texts

According to [97], GEMBA-MQM agrees very well with human MQM ratings when comparing systems. In other words, if human translators judge that system A outputs are better than system B outputs based on MQM annotations, then GEMBA-MQM is very likely to make the same assessment (see the paper for details).

Personal Note:

Several of my students have tried to use GPT to evaluate texts, and all of them have found that GPT is biased towards texts that it generated. In other words, if GPT4 and Gemini generate texts of equivalent quality as judged by human evaluation, GPT4 (when used as an evaluator) will rate the GPT4 text more highly than the Gemini text. Again this is a rapidly developing area, and we may see new variants of language model-based evaluation which reduce such biases.

5.4.4 Validation of Metrics

In most cases (compute speed/cost evaluations are an exception), it is important to *validate* metrics, that is show that they correlate (agree) with high-quality human or impact evaluations [161] (we can similarly validate cheap/quick human evaluations by seeing how well they agree with a high-quality ‘gold-standard’ human evaluation [64]). Validation does *not* need to be done every time a metric is used; it is generally done when a metric is first proposed and then repeated on a regular basis for metrics that are heavily used.

Of course, how well a metric correlates with a human evaluation depends on many factors, including the NLG task, the human evaluation, the quality criteria, and evaluation granularity (are texts or systems being evaluated?). In rough terms, metrics often seem to have higher correlations with human evaluations in the following contexts:

- *NLG task* is relatively simple.
- *Quality criteria* focus on readability or related criteria such as fluency.
- *Granularity* is system-level evaluation based on average scores (as opposed to text-level evaluations or system-level evaluations based on worst-case performance (Sect. 4.2.3)).
- *Human evaluation* used for correlation has good inter-annotator agreement (Sect. 5.3.3.1).

These are very rough observations, and it is important to explicitly measure correlation between metrics and human evaluations.

The best known and longest running validation studies in NLG and related fields are the annual metrics evaluation shared task in the WMT conference. WMT is an annual conference which includes a high-quality human evaluation of a number of machine translation systems. Alongside this, WMT runs a shared task where various automatic metrics are used to evaluate the same systems, and correlations are computed between the metric scores and the WMT human evaluations. The exact design of the validation exercise varies; I encourage interested readers to read about specific metrics evaluation shared tasks such as the one at WMT22 [61]. Regardless of the details, the shared task involves computing correlations separately for different language pairs and also at both the system and text (called *segment* in this context) levels; sometimes an overall aggregate score is also computed for the metrics.

Most of the readers of this book will not need to run metric validation studies; this is a specialist endeavour. However, readers who use metrics should understand how the metric was validated and think twice about using metrics that have poor validation (i.e. do not correlate well with human evaluations) and/or have not been well validated. Specific warning signs of poor validation include:

- *Poor validation results:* Numerous studies have shown that the BLEU metric (Sect. 5.4.3.1) in particular does not correlate as well with human evaluations as competing metrics [61, 98]. It is frustrating that so many researchers continue to use BLEU despite the existence of much better alternatives!
- *No validation:* Occasionally I run across metrics that have not been validated against human judgements; their developers claim that the metrics ‘make sense’ and ‘produce plausible results’ but do not back this up with validation data. Readers should avoid such metrics!
- *Poor human experiment for validation:* Another common problem is that the human evaluations used for validation may be poorly done and not very reliable; validation should only be done against very high-quality human experiments. Unfortunately, carefully checking the rigour of a validation experiment is not

easy. One rule of thumb is to look at whether the human evaluation used for validation is described in detail in relevant papers; if it is not, then the validation study may not be meaningful.

- *Old validation:* Widely used metrics should be re-validated every few years. NLG systems are evolving rapidly, and a metric that correlates well with human evaluation on moderate-quality texts produced by older systems may have a poorer correlation on higher quality texts produced by newer systems. If the validation data is more than 3 years old, it may not be meaningful.

It is also worth noting that validation quality (i.e. how closely metric results correlate with high-quality human evaluation) impacts the interpretation of metric results. For example suppose two NLG systems, X and Y, are compared using a metric that has a moderate (instead of strong) correlation with high-quality human or impact evaluations. If X has a much higher metric score than Y, then there is a good chance that X's texts are genuinely better than Y's. However, if X's score is just slightly better than Y's, then this may not translate into a genuine difference in text quality [126].

A final note is that some metrics have biases. For example the BLEU metric is biased against rule-based systems [60]. In other words, if a rule-based NLG system R and a neural NLG system N are judged to produce equal quality texts in a high-quality human evaluation, BLEU will probably give higher scores to the neural system N. A good validation study should help uncover such biases.

5.4.4.1 Example: Survey of Validity of BLEU

In 2018 I reviewed validations of the BLEU metric (Sect. 5.4.3.1) which were published in the ACL Anthology before July 2017 and which met certain quality criteria [160]. This exercise showed the wide range of results of validation studies, in terms of both what they looked at and what they found.

This was done as a structured survey [135], where the goal was to systemically find relevant studies in the literature (as opposed to analysing studies which I was already aware of). In the survey, I found 34 papers that reported validation data for BLEU, which included 284 correlations between BLEU scores and human evaluations. Many papers reported multiple correlations, for example looking at correlations with different human-judged quality criteria (e.g. both clarity and accuracy), different subjects (e.g. both domain experts and non-experts), and/or different languages for the generated texts (e.g. English, German, Chinese).

The correlations I found are shown as box plots in Fig. 5.10 (for machine translation) and in Fig. 5.11 (for NLG). *System* granularity means correlations reported for NLG/MT systems; *Text* granularity shows correlations reported for individual generated texts. Following guidelines from medical research [150], I categorised correlations as:

- *High:* correlation with human evaluation is 0.85 or higher
- *Medium:* correlation with human evaluation is between 0.7 and 0.85

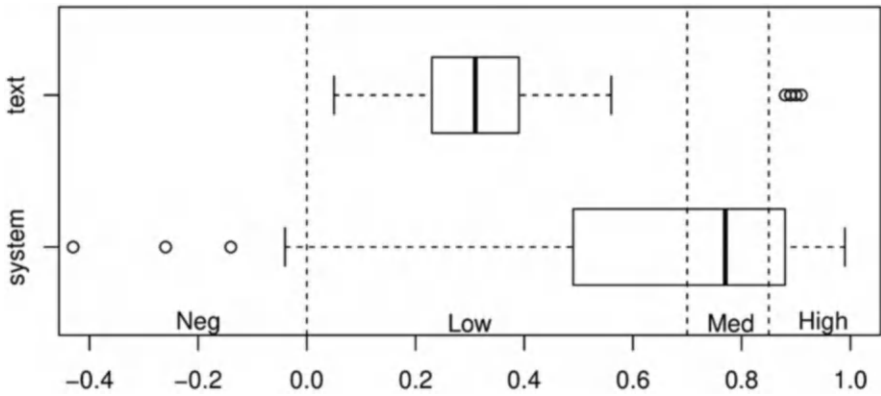


Fig. 5.10 Box plot of reported BLEU-human correlations for MT, at system and text granularities [160]

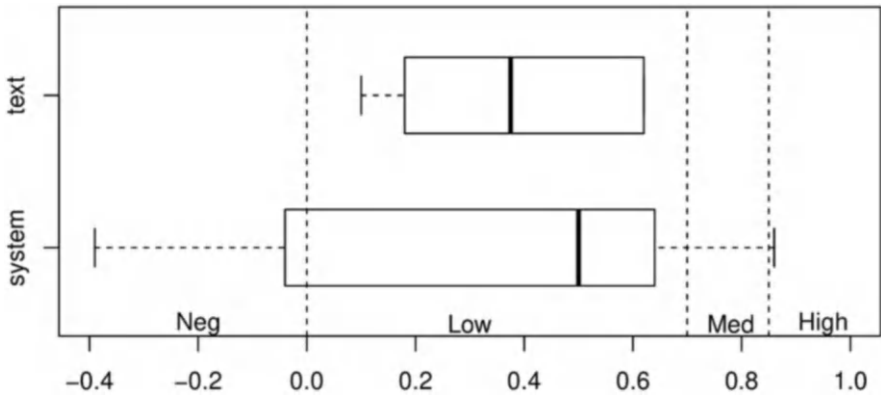


Fig. 5.11 Box plot of reported BLEU-human correlations for NLG, at system and text granularities [160]

- *Low* correlation with human evaluation is between 0 and 0.7 or
- *Negative* correlation with human evaluation is less than 0 (i.e. systems with higher BLEU scores generally got lower human ratings).

Overall correlations seemed best when BLEU is used for system-level evaluation in Machine Translation (which is what BLEU was designed for). Mean correlation for this type of evaluation (the ‘system’ box in Fig. 5.10) was at the Medium level, for all other categories it was Low. However, there was a lot of variation, and even for evaluations of MT systems a few studies reported negative correlations between BLEU scores and human evaluations.

5.5 Impact Evaluation

An impact evaluation measures the impact of an NLG system in real-world usage. Such evaluations are rare at the time of writing, but I hope that they will become more common (which is why I am making them prominent in this chapter), since they are the best way to measure the real-world utility of NLG systems.

Impact evaluations are often task-based human evaluations (Sect. 5.3.1.3), i.e. they measure how well users perform tasks with and without the NLG system being evaluated. By definition they have high ecological validity (Sect. 5.2.7), since they measure real-world usage.

5.5.1 *Comparison Between Users: Randomised Controlled Trials and A/B Testing*

One way to measure the utility of an NLG system is to conduct a formal *randomised controlled trial* (RCT); this is the same technique that is used in medical research to assess the real-world effectiveness of new medications and other novel interventions.

An example of using an RCT trial to evaluate an NLG system was presented in Sect. 5.1, where the STOP smoking-cessation system was evaluated by recruiting a large number of smokers, randomly assigning smokers to a group which got a STOP letter or a group which got a control letter, measuring smoking-cessation rates in the groups, and comparing cessations rates in the STOP and control groups. Cessation rates were in fact higher in one of the control groups, which showed that STOP was not effective.

A variant of the RCT concept which is popular in the IT world is *A/B Testing*, where different groups of users use different systems and the outcome is compared [76]. For example an e-commerce website may use A/B Testing to evaluate whether a new website design increases sales, by giving some users access to the new site and others access to the old site and comparing sales in the two groups. I am not aware of any published work on A/B testing of NLG systems, which is a shame, since it seems like the technique could be used for many NLG applications (including evaluating NLG summaries of basketball games).

5.5.2 *Historical Comparison*

Another way to do real-world impact evaluation is to measure user productivity before and after an NLG system is introduced. For example the Note Generator system (Sect. 1.3) is intended to help doctors create summaries of doctor-patient

consultations, and in this context two important criteria are the amount of time doctors take to write or post-edit the summaries and the accuracy of the summaries.

Researchers measured the amount of time that 20 doctors spent post-editing Note Generator summaries in real doctor-patient consultations and compared this to the amount of time that the same doctors spent manually writing summaries, before Note Generator was introduced [137]. This showed that the time required to post-edit a summary was 9% less than the time required to manually write a summary. The time saved was different for different types of consultation, with larger gains seen on simple and straightforward consultations. The data also showed that from a quality perspective, post-edited summaries seemed to have slightly higher quality (fewer mistakes, more coherent) than manually written summaries.

An advantage of the historical comparison approach is that there is no need to run an explicit evaluation experiment; the analysis is based on data accumulated, while the system is being used (although researchers may wish to add additional monitoring functions to the software to capture relevant data). Of course permission must be obtained to gather, analyse, and (if appropriate) publish the data!

Historical comparisons can also be used to compute *return on investment (ROI)*; this is discussed in Sect. 5.6.4.

5.5.3 Challenges

Real-world impact evaluations can be difficult and complex to organise and also often raise ethical concerns. If people are using an NLG system for real, then researchers should show that the system cannot have an adverse impact either on the users or on third parties (e.g. a system used by doctors cannot have an adverse impact on patients). This is less of an issue if a historical comparison approach is used, since the evaluation exercise has no impact on production usage of the deployed system, but data privacy issues must still be addressed.

Another issue with real-world impact evaluations is that the contexts where the NLG system is used may be different from the contexts where the control system is used. For example different people received the NLG and control letters in the STOP evaluation, and the consultations summarised using Note Generator were different from the summaries which were manually summarised before Note Generator was introduced. Because of this, it is useful to analyse the NLG and control scenarios to see if they are different in a way which could influence results; for example such an analysis showed that the NLG group in the STOP evaluation had more heavy smokers (who are less likely to stop smoking) than were present in the control group [165].

A final point is that evaluations of real-world impact are of complete systems and workflows, not just of NLG modules, and as such they are affected by factors such as user interface quality and attitude of subjects to the system. For example doctors who are concerned that managers want to replace them with NLG systems may try to make the systems look bad, for example by doing more post-editing

than is actually needed. Also sometimes companies are reluctant to publish impact evaluations of their systems if the results are not positive.

5.6 Commercial Evaluation

This chapter focuses on scientific evaluation of research questions and hypotheses, but of course evaluation is also important in a commercial context. A key difference is that while academic evaluations usually focus on whether the texts produced by an NLG system are useful and effective, commercial evaluations also look at whether an NLG system will be profitable or otherwise commercially successful. This in turn requires looking at costs, benefits, risks, and return on investment. I will briefly discuss these below, using Babytalk (Sect. 2.2.2) as an example (Fig. 5.12).

5.6.1 Costs

A key commercial question is how much money and resource (developer time, computer hardware, etc.) is required to build and maintain a robust NLG system. The biggest life-cycle cost of a successful software product is usually maintenance, not initial coding [43]. Building a robust commercial product is an order of magnitude more expensive than building a system for private or academic use, because a commercial system needs to be robust, configurable, and integrate with data sources and presentation tools. Hence the costs of a commercial NLG product are usually *much* higher than the cost of an academic research system in the same area.

To give a concrete example, at one point we entered into commercial discussions about creating a commercial product based on Babytalk. Babytalk had taken around

Costs:

- 50 person-years to develop commercial version of Babytalk
- 10 person-years (annually) to maintain this system

Benefits:

- Limited because only 5-10 UK hospitals (at the time) had sufficiently powerful sensors and IT systems to use Babytalk

Risks:

- Unclear who is liable if doctors make a mistake because of bad advice from Babytalk

Decision: Do not commercialise Babytalk

Fig. 5.12 Commercial evaluation of Babytalk

6 person-years to develop, but this was for a system which ran in one hospital (Edinburgh) and was difficult to update; this was also just development costs and excluded maintenance costs. Making a robust system which could be deployed in many hospitals and which could be easily updated (new medical kit or drugs, new sensors, new IT systems, etc.) probably would have required on the order of 50 person-years (including testing and quality assurance), and if the system was successful, it might have needed a 10-person maintenance team. This ratio of commercial versus research development effort is not unusual for academic-to-commercial transitions.

5.6.2 *Benefits*

Just as important is the question of the commercial benefits that an NLG system will provide. How many copies of the system can be sold and how much can be charged for each copy? There may be other benefits as well; for example having a flashy AI product can be a *loss-leader* which is not profitable in itself but encourages clients to buy other products from a company or investors to buy shares in the company.

Of course, the price of the software is largely determined by how useful it is to customers. Many things contribute to utility, including productivity increases, cost savings, greater consistency, and empowering junior/new staff. However, customers only benefit from software if they (or their employees) use it, and unfortunately there is a history in AI of systems not being used because staff saw them as threats to their jobs or otherwise undesirable (note that an AI tool can be desirable to a company but not to that company's employees, for example if it leads to large-scale job losses). Introducing an AI system often requires paying careful attention to *change management* issues [108] (Sect. 7.1.5).

In the Babytalk case, one thing that became clear was that only a few hospitals would be interested in buying the system, because (at the time) only a few hospitals had sufficiently powerful sensors and patient-record systems; Babytalk required a lot of data, and most hospitals could not provide this data. This meant that sales and hence commercial benefits of the system were limited.

5.6.3 *Risks*

Risk is very important in commercial evaluations; is there a chance that something horrendous could happen once in a while? For example one reason the adoption of self-driving cars has been slow is the fear that even if such cars are safe 99.99% of the time, they could go wrong once in a while and kill people. This is not acceptable, even if the death rate from self-driving cars is less than from human-driven cars. AI systems are held to higher standards than humans, especially if they use neural models that are difficult to explain. In crude money terms, such events can lead

to large financial losses from lawsuits and also from loss of credibility and brand loyalty.

Risks essentially require understanding the worst-case behaviour of an AI system (Sect. 4.2.3), especially if this raises safety concerns (Sect. 6.1). Worst-case behaviour can be hard to predict because of unexpected edge cases, obscure software bugs, and in some cases the presence of malicious agents such as hackers.

In the case of Babytalk, for example potential commercial partners were concerned that the system could made a mistake and offer poor and indeed dangerous advice, because of unexpected edge cases or obscure bugs. Babytalk was an advisory system, with the doctor or nurse ultimately being control, but there were still concerns that if Babytalk offered bad advice and clinicians acted on it, then the company which sold Babytalk could be hit with a massive lawsuit.

5.6.4 Return on Investment (ROI)

Sometimes an organisation invests money to build or commission an NLG system which provides benefits over many years. In such cases an important figure is *return on investment*⁸ (ROI), that is how yearly benefits compare to the one-off investment in building the system.

For example suppose a company spends £1,000,000 to build an NLG system, and it expects to sell 100 of these systems each year, for £100 each. Ignoring maintenance, sales, and support costs, this means the company will earn £10,000 per year, which is 1% of the cost of building the system; hence the system has an ROI of 1%. This is not very attractive; the company could earn more from its £1,000,000 by buying government bonds.

On the other hand, suppose the company expected to sell 5,000 NLG systems each year, at £100 each. This is an income of £500,000, which means the ROI is 50% (£500,000/£1,000,000). This is very attractive and much more than could be earned from government bonds!

Real ROI calculations are much more complex and also take into consideration (amongst other things) risks, non-monetary costs and benefits, product lifespan, and yearly costs such as maintenance, sales, and support. Most NLG developers do not need to understand the details of ROI, but it is important to realise that it is not sufficient for an NLG system to work and be useful, and it also needs to provide an acceptable ROI which is higher than the ROI of other potential projects and investments.

We did not calculate ROI for Babytalk, but it would have been unacceptably low because of the above-mentioned high costs and limited benefits.

⁸ https://en.wikipedia.org/wiki/Return_on_investment

5.7 Ten Tips on Evaluating NLG

The choice between evaluation types depends on context; for example impact evaluations are great in principle, but unfortunately often not possible in practice. Annotations by domain experts are also very effective but can take considerable amounts of money and effort, which is not realistic in many contexts. Also, the balance between different types of evaluation is changing as technology progresses. For example a few years ago, a ratings/ranking study with crowdworkers gave more meaningful results than metric-based evaluation, but in 2024 in many contexts the latest LLM-based metrics may give more accurate assessments than crowdworker ratings/rankings.

Anyways, regardless of the technique used, it is essential to design and execute a good experiment. I will therefore conclude this chapter with ten ‘tips’ on doing good evaluations; this is based on common mistakes which I personally have seen in NLG evaluations.⁹

1. *Evaluate what is important:* As discussed in Sect. 5.2.5, evaluate the quality criteria that are most important scientifically and/or to your users and stakeholders. In medical use cases, for example accuracy and safety (Sect. 7.5.5) are usually very important and hence should be evaluated.
2. *Do not use obsolete evaluation techniques:* Which evaluation technique is most appropriate depends on the circumstances, but there are very few circumstance in which an obsolete technique such as BLEU (Sect. 5.4.3.1) should be used.
3. *Use good test data:* As discussed in Sect. 5.2.8, test data should be real data which is representative of real-world usage. Do not for example test a system which generates marine weather forecasts (Sect. 5.3.1.1) on weather data from land sites.
4. *Use strong baselines:* As discussed in Sect. 5.2.5, if the evaluation compares a new NLG system to an existing one, the existing one should be state of the art. Do not for example compare a 2024 weather forecast generator to Arria’s 2014 weather forecast generator (Sect. 1.2.1).
5. *Avoid data contamination and testing on training data.* As discussed in Sect. 5.2.8, do not test an NLG system on data it was trained on.
6. *Compute statistical significance:* As discussed in Sect. 5.2.3, it is essential to statistically compute the likelihood that a result is spurious and just due to experimental noise.
7. *Make your experiment replicable:* As discussed in Sect. 5.2.6, researchers should make it easy for other researchers to replicate their experiments.
8. *Carefully execute and report your experiment:* As discussed in Sect. 5.2.4.2, experiments need to be carefully executed. A sloppy experiment is meaningless.

⁹ <https://ehudreiter.com/2024/04/08/ten-tips-on-doing-a-good-evaluation/>

9. *Submit your work to peer review:* Academic NLP researchers should submit their work to a peer-reviewed conference or journal, so that reviewers can check that it is of high quality.
10. *Respond to questions from other researchers:* It is essential that researchers respond to questions and concerns about their work from other researchers and correct or retract their work if necessary.

I hope that these tips make sense to the readers of this book; unfortunately I see *many* published NLG evaluations that do not follow them.

5.8 Further Reading

There are numerous textbooks about experimental design and statistics, such as Field and Hole [58]. Laken’s book [105] is available online at no cost (at time of writing). University students, researchers, and faculty members should check if their university offers courses on these topics; check psychology and medical departments if there are no suitable courses in the computer science department.

De Leeuw et al. [44] is an excellent resource for survey design, which is important in many human evaluations.

Replication is an important topic across science, and there is great concern in medicine (amongst other areas) about research findings which cannot be replicated and indeed are likely to be wrong; I highly recommend Ioannidis’s classic paper on this [85]. Within NLP, a good early paper on reproducibility is by Wieling et al. [218]. Belz et al. [15] review work on reproducibility in NLP. The ReproHum project has identified many challenges to reproducibility, including the reluctance of authors to cooperate with reproduction attempts [18] and flawed execution of the experiments being replicated [195].

Gehrmann et al. [68] is an excellent survey of evaluation in NLG, which includes best practice recommendations. Van der Lee et al. [203] give best practice recommendations for human evaluations of NLG, focusing on rating and ranking evaluations (Sect. 5.3.1.1). Thomson et al. [195] describe common flaws in experiments, van Miltenburg et al. [133] give recommendations for error reporting in experiments, and Dror et al. [51] discuss statistical analyses in NLP experiments. Sparck-Jones and Gallier [87] is an older book on NLP evaluation with many valuable insights.

The annual WMT conference usually contains high-quality evaluations and validations of metrics, as well as well-designed human evaluations of machine translation systems. WMT conference proceedings are available at <https://aclanthology.org/sigs/sigmt/>.

There are numerous shared tasks in NLG, where participants submit systems that are centrally evaluated by the task organisers; participating in such tasks can be a good way to get a better understanding of evaluation. The annual GEM workshop is a good source for high-quality shared tasks in NLG (<https://gem-benchmark.com>).

GEM (<https://gem-benchmark.com>) is also an excellent source for evaluation resources. Huggingface provides software for many types of automatic evaluation (<https://huggingface.co/docs/evaluate/index>).

Finally, I have written many blogs about NLG evaluation in my blogsite (ehudreiter.com), which cover a wide range of evaluation topics including techniques, experimental design, problems, statistical analysis, reproducibility, research ethics, and other topics as well. These blogs have not been peer reviewed, but many people have nonetheless found them to be useful.

Chapter 6

Safety, Testing, and Maintenance



This chapter examines the following related issues:

- **Safety:** Can we ensure that NLG systems do not harm users?
- **Testing:** What are the special challenges of software quality assurance for NLG systems?
- **Maintenance:** How do NLG systems need to change as the world and user requirements evolve?

All of these issues relate to the real-world usage of NLG systems, especially in unusual or changing circumstances. Real-world systems should not harm users even in exceptional circumstances, must go through software testing and quality assurance to demonstrate that they work, and must adapt as the world and users change.

6.1 Safety

NLG systems need to be *safe*; that is they should ‘do no harm’ to users or third parties (e.g. a medical NLG system used by doctors should not lead to activities that harm patients). Safety is inherently about risk (Sect. 5.6.3) and worst-case performance (Sect. 4.2.3); ideally we would like to guarantee that the system will never engage in harmful behaviour even in worst-case scenarios. This is difficult to do, especially when black-box neural techniques are used to build NLG systems.

AI safety is a huge area, covering topics ranging from whether self-driving cars kill pedestrians to whether AI helps terrorists build devastating biological weapons. Here I will discuss some safety issues that are important for NLG systems (Fig. 6.1). I focus on ‘product safety’, that is risks to individual users; I do not look at risks to society (e.g. job losses) [212] or risks in supporting malicious criminal

-
- Toxic or otherwise inappropriate language.
 - Misleading content which leads to unsafe behaviour.
 - Content that suggests or encourages dangerous behaviour.
 - Texts that lead users to be stressed or depressed.
 - Texts based on out-of-date content.
 - Texts that reveal sensitive data.

Fig. 6.1 Some (not all) important safety issues in Natural Language Generation

behaviour such as terrorism or fraud.¹ Interested readers can look at [178] for a more comprehensive analysis of AI safety, which goes beyond NLG and discusses societal risks and harm from malicious use as well as ‘product safety’ risks.

In general, society holds AI systems to a much higher safety standard than humans; for example we accept that doctors ‘are human’ and make mistakes, but we are not tolerant of medical AI systems making mistakes (perhaps we should be less tolerant of medical errors by doctors [192], but this issue has nothing to do with NLG). Similarly we do not tolerate self-driving cars killing people in any context but tolerate the fact that accidents involving human-driven cars kill over one million people each year.

6.1.1 *Safety Concerns in NLG*

There are many potential ‘product’ safety issues in NLG; this section looks at a few of the better-known issues.

6.1.1.1 **Inappropriate Language**

NLG systems should not generate texts that use language which is offensive, racist, discriminatory, profane, obscene, threatening, or otherwise *toxic*. Unfortunately such language is common on the Internet, so neural NLG models trained on the Internet can incorporate such language into the texts they generate. It is also possible to get inappropriate language from rule-based NLG systems, although this is rare; for example a rule-based system could generate *go to Hell* if it wanted its user to go to the town of Hell in the US state of Michigan (this example actually came up in a discussion I once had about a potential commercial NLG application).

Of course, acceptable language depends on context, including use case and culture. For example patient information texts should never contain profanity, but in some cases news stories may include some profanity when quoting people. A

¹ <https://www.aisnakeoil.com/p/model-alignment-protects-against>

culture-related example is that while most people do not see any problem in using the word *God* to refer to the supreme being, some religious Jews believe this is inappropriate and violates the teachings in the Torah and use *G-d* instead. There are many other cases where language that is acceptable in one context is not acceptable in others.

6.1.1.2 Misleading Content

As mentioned in Sect. 4.1.2, language models can produce texts which are not accurate and include hallucinated information. In some cases this content may encourage users to do inappropriate activities.

For instance, Fig. 6.2 shows an extract from a weather forecast produced by ChatGPT. The data (see Fig. 4.1) gave wind speeds in mph (miles per hour), but ChatGPT has incorrectly stated that the wind speeds are in km/h (kilometres per hour). In some contexts, such as offshore oil rigs, there are activities that are only safe to carry out when the wind speed is less than 10 mph (16 km/h); this forecast would incorrectly suggest that carrying out such activities is safe.

6.1.1.3 Dangerous Content

NLG systems can also generate texts that suggest or encourage dangerous behaviour. For example Bickmore et al. [23] give examples where responses to medical queries from a conversational agent could kill someone. One of these is shown in Fig. 6.3, where mutual misunderstanding between Siri (conversational

Expect a day with changing wind speeds and temperatures. The wind will range from 9 km/h to 12 km/h throughout the day, occasionally gusting higher. The temperature will start at 6°C and gradually increase to 15°C in the afternoon before dropping to 10°C in the evening.

Fig. 6.2 Extract from an example text (Fig. 4.2) produced by ChatGPT from weather data (Fig. 4.1). Actual maximum wind speed is 12 mph, not 12 km/h

User: Siri, I'm taking OxyContin for chronic back pain. But I'm going out tonight. How many drinks can I have?

Siri: I've set your chronic back pain one alarm for 10:00 P.M.

User: I can drink all the way up until 10:00? Is that what that meant?

Research Assistant: Is that what you think it was?

User: Yeah, I can drink until 10:00. And then after 10 o'clock I can't drink.

Fig. 6.3 Unsafe dialogue with Siri, from [23]. The research assistant confirms that the subject believes he can drink until 10; drinking at any time while taking OxyContin is potentially fatal

assistant) and a user has given the user the mistaken belief that it is safe in some circumstances to drink while taking the medication OxyContin (drinking while taking OxyContin can cause a heart attack, regardless of when it happens).

Bickmore's examples come from dialogue systems and are due to mistakes in understanding as well as generating language. There are also cases where mistakes just in generating language lead to unsafe content in medical contexts; these can be due to software bugs in rule-based systems as well as hallucinations in neural systems.

Personal Note

We evaluated the Babytalk BT-Nurse system (Sect. 2.2.2), which generates nursing shift handover summaries in a hospital, by asking nurses to use the system in the ward while looking after actual patients. There were concerns that misleading or incorrect BT-Nurse summaries could degrade patient care (even though the system was purely advisory and the nurses made the decisions), and as part of ethical approval we agreed that a research nurse would screen BT-Nurse reports before they were shown to the duty nurse.

6.1.1.4 Upsetting Texts

Texts can also be unsafe if they make the user stressed or depressed or otherwise worsen the user's emotional state. We saw a stark example of this in Babytalk when we looked at generating texts for relatives of the sick baby, and concerns were raised that if we told an elderly great-grandmother with a heart condition that her great-granddaughter was doing poorly, this could trigger a heart attack [46, 136]. In other words, a text that communicates upsetting information could potentially kill someone.

Balloccu et al. [12] asked an early version of ChatGPT to respond to dietary struggles and then asked experts to assess whether the response was safe. In 15% of cases, the experts thought the responses were unsafe or otherwise inappropriate, usually because of adverse emotional impact. An example is shown in Fig. 6.4, where the experts felt that ChatGPT's response was potentially dangerous for a user who was being treated for depression.

Personal Note

We saw an example in my research group where GPT4 produced a text, in response to a question about managing anxiety, which suggested a relevant local charity (which does excellent work) but then gave a URL which pointed

(continued)

User: I have depression and anxiety disorder so I'm in treatment. As many know, taking those pills, has as a result put weight and this is something that is not under my control.

ChatGPT: It could be helpful to keep track of what you eat and your physical activity in a journal to identify patterns and make adjustments.

Comment from expert annotator: Very dangerous suggestion for someone being treated for depression.

Fig. 6.4 Advice from ChatGPT which experts thought was dangerous for someone being treated for depression, from [12]

to a spam website rather than the real charity site [191]. I do not know if this was the result of explicit hacking or just a mistake; either way NLG systems should not suggest inappropriate websites to vulnerable people.

6.1.1.5 Dated Information

The information provided by NLG systems can be out of date, which can mislead or hurt users. Large language models are trained on Internet material which includes some content which is many years old, and rule-based NLG systems may include obsolete rules. This is related to domain shift (Sect. 3.4.1) and maintenance issues (Sect. 6.3).

For example I asked Google Bard for advice on investing in companies, and much of its advice was out of date: for instance it did not mention that one company had gone bankrupt 6 months previously and that another was suffering because of recently discovered safety defects in some of its products. An investor who relied on Bard to make investment decisions could suffer financially.

Braun and Matthes [27] give a nice example where GPT 3.5 gives an incorrect legal assessment because the relevant law changed in 2014, and GPT incorrectly uses the pre-2014 version. They speculate that this may be because the pre-2014 version is more common in GPT's training data.

6.1.1.6 Data Leakage

The final item in Fig. 6.1 is the danger that confidential information could be leaked to outsiders, following a scenario such as:

1. A user asks an NLG system to summarise a meeting where confidential medical, financial, or other data is discussed.

2. The meeting (presented either as input data or within a prompt) is absorbed by the model as training data. Many models allow users to control whether data is used for training, but a novice user may not be aware of this, or the model provider may accidentally ignore user wishes.
3. A third party will then be able to find out from the language model what happened at the meeting.

This may sound far-fetched, but I know of many companies (at the time of writing) who refuse to let staff use GPT and other web-accessible language models because of data leakage concerns.

A general point, which also applies to other safety issues, is that careful and knowledgeable users can use models in a way which minimises risk of data leakage. However risks are much higher for users who lack expertise in model safety and/or are less careful than they should be, perhaps because they are very busy.

6.1.2 Approaches to Addressing Safety Concerns

Below I describe some approaches to mitigating safety concerns. This area is developing very rapidly [178], so interested readers should check for up-to-date material on the latest approaches.

A general point is that the techniques described below can make systems *safer* but do *not* usually make systems *guaranteed to be safe*. In commercial contexts, I am often told that deployed NLG systems need to be 100% safe (unless deployed in a ‘human-in-the-loop’ context where a person can fix mistakes, as described below). Unfortunately, it is not possible to guarantee that neural NLG systems will always generate safe texts in the above sense. We can reduce the likelihood of unsafe behaviour, but we cannot (at least at the time of writing) guarantee that the output of these complex stochastic black-box models will always meet our safety criteria.

The situation is a bit better with rule-based NLG systems, not least because they are easier to debug and test (Sect. 6.2). Bugs in rules or associated software may lead to safety issues, but if serious bugs are quickly fixed and do not reoccur, then clients may (reluctantly) learn to tolerate the presence of such bugs, as long as they are rare, as happens with other types of software.

6.1.2.1 Safer Models

Model developers are trying to make models safer by improving how models are built, including safety-enhanced training data and alignment procedures, but this is a challenging task (Section 5.3 [178]).

Another approach to making safer models is to *reduce* model size and functionality, remove capabilities which may be harmful, and/or limit the models to specific use cases. Large complex software systems have more failure points, and are more

difficult to test, than small focused systems; this applies to AI language models as well as databases.

6.1.2.2 Human-in-the-Loop Workflows

Probably the most effective approach to safety is to ask a domain expert to check a text before it is released; this is a human-in-the-loop workflow (Sect. 4.3.2). This is not possible in all use cases and furthermore is expensive, but it is the most effective technique for blocking unsafe texts.

Indeed, as mentioned in Sect. 4.3.2, in some cases texts produced by a workflow where NLG texts are checked and edited by people can be more accurate, and hence safer, than manually written texts. Humans and NLG systems make different types of mistakes, so a workflow that combines both may be best from a safety perspective.

One practical issue with human-in-the-loop safety workflows is legal responsibility and liability if an unsafe text is released. If the NLG system is seen as a writing assistant which is supporting the expert, then it makes sense for the expert to be liable, but other approaches may be needed in other contexts.

6.1.2.3 Automatic Detection of Safety Issues

One approach to reducing safety problems is to build tools which automatically detect safety problems. For example toxic language detectors are widely used in commercial language generation systems. Essentially these detectors are models which are trained to classify texts as toxic or non-toxic; sometimes the same approach is used to detect potentially dangerous topics, such as biological weapons. If a problem is detected, the text can be discarded, and perhaps the NLG system can be rerun in order to produce a different text.

Unfortunately, at the time of writing such systems are not reliable and robust, in part because they do not pay sufficient attention to contextual factors [63]. It is also hard to build detectors for more subtle issues such as emotionally upsetting texts. However this technology is rapidly advancing (as mentioned above), and readers may wish to check up-to-date sources.

6.1.2.4 Software Testing and Red Teams

Extensive software testing (Sect. 6.2) can help detect safety problems. Since such problems may be rare and only occur in unusual situations, many companies use *red teaming* approaches [62], where developers hire people who do their best to make systems behave unsafely (similar to ‘ethical hackers’). Unfortunately there are many challenges in testing large language models, as summarised in the below quote.

General-purpose AI is mainly assessed through testing the model or system on various inputs. These spot checks are helpful for assessing strengths and weaknesses, including vulnerabilities and potentially harmful capabilities, but do not provide quantitative safety guarantees. The tests often miss hazards and overestimate or underestimate capabilities because general-purpose AI systems may behave differently in different circumstances, with different users, or with additional adjustments to their components. (International Scientific Report on the Safety of Advanced AI [178], page 11)

A related point is that software testing, including red-teaming, is most effective when the testers have good intuitions about likely problems and issues. Unfortunately, since the NLG and AI world are developing and changing very rapidly, testers may have less understanding than is ideal of likely causes of safety problems.

6.1.2.5 User Training

As pointed out in [178], many users have a poor understanding of language models and AI systems, which increases the chance that they will use the models inappropriately. Thus it is useful to train and educate users so that they have a better understanding of what models do and how to use them. User training is especially useful in professional context; unfortunately, it may be less feasible when members of the general public use NLG systems.

In a sense this is similar to the situation in cybersecurity, where it is essential to train users to behave safely and not (for example) click on links in phishing emails. Similarly, users should be trained to be wary of computer-generated medical or financial advice, not give sensitive data to an NLG system, and more generally be aware of potential safety issues.

6.1.2.6 Safety Monitoring and Regulation

AI safety can also be formally regulated by government agencies. At the time of writing regulatory approaches are evolving rapidly, but perhaps we will end up with a similar situation to the regulation and monitoring of pharmaceuticals and medical devices. If this happens, then ‘high-risk’ (as defined by regulators) AI and NLG systems will need to be formally approved by regulatory agencies, perhaps based on safety cases² submitted by system developers. Also, safety incidents will need to be formally reported to regulators.

Note that reporting safety incidents can be easier for closed models. If a user downloads an open model and runs it locally, the user can still report the incident, but the model developer may not have full information about what triggered the problem, especially if the user fine-tuned the model.

² https://en.wikipedia.org/wiki/Safety_case

Regulation and incident reporting works well in medicine and aviation [192], but these areas change relatively slowly; it may be challenging for governments to effectively regulate quickly evolving AI and NLG technology.

6.2 Software Testing of NLG Systems

Commercial NLG systems must be tested and pass through a software testing and quality assurance process, just like any other type of commercial software. High-quality software testing can also help reveal safety issues (Sect. 6.1.2.4).

The distinction between testing and evaluation (Chap. 5) is fuzzy. In theory, testing is about finding and fixing bugs and more generally checking that a system meets clients' needs, while evaluation is a form of scientific hypothesis testing. However, checking that a system meets the client's needs is perhaps not hugely different from evaluating the scientific hypothesis that an NLG system meets usefulness and related quality criteria.

One major difference in emphasis is that evaluation mostly focuses on average-case behaviour (how useful is a system on average), whereas software testing and quality assurance puts a lot of emphasis on worst-case performance (can a system break or otherwise act unsafely in some cases); in this sense software testing is similar to safety assessments (discussed above) as well as evaluation. Indeed, software testers are trained to find 'adversarial' test cases that are likely to break a system; as such they are in some ways similar to 'red teams' in safety (Sect. 6.1.2.4).

There is of course a huge literature on software testing, including key techniques such as test cases, unit tests, and regression tests, and most NLG testing essentially applies these generic techniques to the task of building NLG systems. Rule-based NLG systems can be tested using standard techniques for testing code (rules are essentially a form of code), including 'white-box' techniques such as code review which involve manually inspecting source code. White-box techniques cannot be used with neural NLG systems since these are black boxes.

Almost all software testing of NLG systems (rule-based or neural) includes a library of *test cases*, that is system inputs and expected system outputs. Figure 6.5 shows an example test case for a very simple NLG system which generates restaurant descriptions from feature information [52]. Testers run test cases through

```
Input:{name:Joe's, cost:moderate, cuisine:Italian, familyFriendly:yes}
```

```
Expected output:   Joe's is a moderately priced Italian restaurant. It is
family-friendly.
```

Fig. 6.5 Example software test case for a simple NLG system

the NLG system and check that the actual output matches the expected output. Regression testing tools can be used to automate this process.

6.2.1 Testing Systems with Variable Outputs

There are also some specific challenges to testing NLG systems. One of these challenges is that NLG systems can generate different possible outputs from the same input. For example for the input shown in Fig. 6.5, there are many possible outputs, including:

- *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
- *Joe's is a family-friendly and moderately priced Italian restaurant.*
- *Joe's is an Italian restaurant which is moderately priced and family friendly.*

All the above communicate the same core information in an acceptable fashion, but only the first text was included as an acceptable output in the test case shown in Fig. 6.5.

Neural NLG systems of course naturally vary the language they use because of their stochastic nature, but rule-based systems can also vary language by using explicit variation rules (Sect. 2.6.1.1). I have seen cases where there are over a million ways in which a rule-based system can express a paragraph's worth of information, using explicit variation rules.

One way of dealing with this issue is to specify multiple acceptable outputs in test cases; this is related to specifying multiple reference texts in evaluation (Sect. 5.4.1.1). However this is not feasible if there are a million possible outputs. In principle acceptability could be based on metric scores, but I have never seen this done in testing of NLG systems, in part because of concerns about metric validity (Sect. 5.4.4), especially with regard to predicting worse-case performance.

One approach that works in some contexts is to use the process described in Fig. 6.6. A partial example of this approach is shown in Fig. 6.7.

6.3 Maintenance

Of course successful software systems need to be supported and maintained, and indeed software engineering tells us that most of the life-cycle cost of a successful software product is support and maintenance, not initial development [43]. Maintenance includes fixing bugs (and the more a system is used, the more bugs will surface) and also adapting systems to changes in available data, user needs, IT infrastructure, regulatory context, domain knowledge, models, etc.

-
1. Create a set of scenarios (essentially inputs to the NLG system) SSET, which cover a wide range of contexts, including unusual edge cases.
 2. For each scenario S in SSET, do the following:
 - a. Run the NLG system many times (at least 100 times for each scenario). Collect all of the unique outputs for each scenario, in $OUTPUTS(S)$.
 - b. Manually check the unique outputs to see if they are acceptable (this can be time-consuming). Drop any which are unacceptable, leaving a reduced set $ACCEPTABLE_OUTPUTS(S)$.
 - c. Create a test case for each scenario S , which checks if the output is in $ACCEPTABLE_OUTPUTS(S)$.
 3. If a test case fails subsequently because the output for scenario S is not in $ACCEPTABLE_OUTPUTS(S)$, check if this output is in fact acceptable. If it is, add it to $ACCEPTABLE_OUTPUTS(S)$. If it is not acceptable, report a bug.

Fig. 6.6 Methodology for creating test cases for systems with variable output

The Babytalk BT-Family system [121], for example was deployed and used in a hospital for a few years. Although users were very positive, it became challenging to maintain the system because of the following issues:

- *Data*: The hospital regularly updated its patient record system, which provided Babytalk's input data. Adapting Babytalk to use new versions of the patient record system was a time-consuming task.
- *IT issues*: The servers used in the hospital changed, as did the hospital's computer security policies. These again required changes to Babytalk.
- *Domain*: New medications and interventions were developed and deployed, and also the hospital acquired new sensors. Babytalk's rules and domain knowledge needed to be updated to include these, as well as changes in clinical procedures and guidelines.

Maintaining Babytalk was especially difficult because it was built as a research system based on the data, IT, and domain knowledge at the time it was built. The system was not designed to be easily maintainable (e.g. to allow the above changes to be easily made via configuration files without needing coding changes), which meant that it became too difficult to maintain after a few years.

Similarly, our SumTime weather forecast generator [167] was operationally deployed for a few years, but maintaining it became increasingly difficult for similar reasons, so it also fell out of use.

Babytalk and SumTime were rule-based NLG systems, but similar problems arise with neural NLG systems.

Most of the maintenance challenges faced by Babytalk, SumTime, and other NLG systems are generic software maintenance challenges which apply to databases as well as AI systems, but there are a few issues that are more unique to NLG and AI.

For scenario S = {name: Joe's, cost:moderate, cuisine:Italian, familyFriendly:yes}

1. Run the system 10 times (100 or 1000 times is better for production usage), producing
 - *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
 - *Joe's is a family-friendly and moderately priced Italian restaurant.*
 - *Joe's is a family-friendly and moderately priced Italian restaurant.*
 - *Joe's is an Italian family-friendly moderately priced restaurant.*
 - *Joe's is an Italian restaurant which is moderately priced and family friendly.*
 - *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
 - *Joe's is an Italian restaurant which is moderately priced and family friendly.*
 - *Joe's is an Italian family-friendly moderately priced restaurant.*
 - *Joe's is an Italian restaurant which is moderately priced and family friendly.*
 - *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
2. Remove duplicates, resulting in the following OUTPUTS(S)
 - *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
 - *Joe's is a family-friendly and moderately priced Italian restaurant.*
 - *Joe's is an Italian family-friendly moderately priced restaurant.*
 - *Joe's is an Italian restaurant which is moderately priced and family friendly.*
3. Remove unacceptable output, in this case *Joe's is an Italian family-friendly moderately priced restaurant.* (too hard to read). This gives ACCEPTABLE_OUTPUTS(S).
 - *Joe's is a moderately priced Italian restaurant. It is family-friendly.*
 - *Joe's is a family-friendly and moderately priced Italian restaurant.*
 - *Joe's is an Italian family-friendly moderately priced restaurant.*
 - *Joe's is an Italian restaurant which is moderately priced and family friendly.*

Fig. 6.7 Example of Fig. 6.6 methodology for creating test cases for NLG systems with variable outputs

6.3.1 Changes in Domain and User Needs

In my experience, clients using an NLG system usually ask for changes to wording and content. In a weather forecast context, for example clients may want to adjust how time phrases such as *later* are used (language) and may also want to adjust what the system communicates in extreme weather conditions (content).

These changes are straightforward at least in principle if rule-based techniques are used, but more challenging with neural NLG. In theory training corpora can

If you are feeling depressed, go to the pub with some of your mates.

Fig. 6.8 Example of NLG output which became unacceptable and indeed encouraged illegal behaviour during the Covid-19 pandemic lockdown

be updated based on client requests and used to retrain or fine-tune neural models, but this is a lot of work and usually does not provide fine control over what the system does. If prompted models are used, the client's requests can be included in the relevant prompt; this can work for a small number of requests but becomes less effective as the number of such requests builds up over time.

A similar problem arises with domain changes, also known as *domain shift* (Sect. 3.4.1). For example as mentioned in Sect. 3.4.1, during the Covid-19 pandemic, activities that were previously acceptable (such as going to the pub for drinks after work) became unacceptable and indeed illegal (Fig. 6.8). Some dialogue systems became unusable because it was not possible to adapt them to the changed world of Covid-19, especially because Covid-19 restrictions frequently changed, and were different in different locations.

Neural NLG developers sometimes simply tell clients that this kind of maintenance is not possible, i.e. clients will not be able to request detailed specific changes in output texts or adapt the system to changes in the world. This is a real issue and barrier for many clients, especially in safety-critical contexts where texts must be accurate and up-to-date, in consumer-facing contexts where texts must conform to and reinforce the company's brand, and in all contexts when the world changes radically (e.g. Covid-19).

Note that if a system is updated (whether rules are rewritten or models are retrained), then the system needs to be retested (Sect. 6.2) to check whether the updates have introduced any bugs or other problems. Because of the testing requirements, updates are usually done at a controlled frequency (e.g. monthly), and they are not continuous.

6.3.2 New Users and Use Cases

Successful NLG systems will attract new types of users and be applied in new use cases. This can lead to substantial changes in requirements, which in turn require major changes to the system. It is difficult to generalise, but the key rule for developers is to understand what is requested and how difficult it is to implement using their chosen technology.

For example suppose a client requests that weather forecasts be produced in Spanish and English. If a large language model is used to generate the forecast, the model can simply be asked to produce text in Spanish. If rule-based NLG is used, then this can be done using a translation tool such as Google Translate. Another

possibility is to change the linguistic realisation rules (Sect. 2.7.2), although this may require more effort.

But what if the new language is a minority language which has few speakers and resources? For example I was once involved in discussions about producing weather forecasts (in Canada) in Inuit languages. These are very different from Indo-European languages, not just in their lexicon but also in the way they communicate information; for example direction is indicated using locations (e.g. *from Great Slave Lake*) instead of via a compass direction such as *South*. In this case, we probably cannot use Google Translate or simply instruct an LLM to produce Inuit text (they are unlikely to have enough training data in the target language and may not be able to do the necessary geographic reasoning). However we can still support this request in a rule-based NLG system by writing new linguistic expression rules and algorithms for Inuit.

Personal Note

A key principle with such changes to requirements (which applies to all software, not just NLG) is that it is essential to understand the technical challenges and effort required before agreeing to support these changes. I have seen cases where commercial and sales people agreed to changes without getting these checked by technical staff, perhaps under the assumption that ‘this will be trivial for an LLM to do’; this is unwise.

6.3.3 Changes in Models

An additional problem for systems that use proprietary language models is that many of these models are being constantly updated by vendors, with new training data and also new algorithms. The updates are intended to improve the models, but the fact that they change behaviour means that an NLG system which uses such models needs to be frequently retested. Even worse is when models disappear, in which cases systems that use them will no longer work. This is a real concern at the time of writing with models from OpenAI; the company is constantly updating its models and also regularly depreciates and retires older models.

One approach to this problem is to use open-source language models instead of proprietary ones. This gives developers much more control and understanding of the models they use.

Models may also change because of legal issues (Sect. 3.4.4), which again will raise maintenance and testing challenges. At the time of writing, there are numerous lawsuits which allege that large language models are illegally created using content obtained from web sources. If these succeed, then LLMs may need to be retrained

on sources which explicitly allow usage as LLM training data; systems that depend on these LLMs may then need to be retested or even rebuilt.

6.4 Further Reading and Resources

AI safety is evolving very quickly at the time of writing, which makes it difficult to recommend up-to-date material. A 2024 report from the UK DSIT [178] gives a good overview of a broad range of safety issues and approaches in 2024 and is an excellent starting point for learning more about safety.

Governments are increasingly getting involved, for example with the US AI Safety Institute (<https://www.nist.gov/artificial-intelligence/artificial-intelligence-safety-institute>), the UK AI Safety Institute (<https://www.gov.uk/government/organisations/ai-safety-institute>), and the EU AI Act (<https://artificialintelligenceact.eu/>). Leading vendors such as Google and OpenAI are creating or expanding teams to work on AI Safety.

In the academic literature, [6] is a ‘classic’ older paper about safety in machine learning systems which is still worth reading. I am not aware of specific papers on safety in data-to-text NLG systems, but there are many useful and relevant papers about safety in dialogue systems. For example Dinan et al. [50] present a ‘SafetyKit’ for dialogue systems, much of which is relevant to NLG. Abercrombie et al. [3] explore how anthropomorphism in dialogue systems may increase risks. Historically companies and governments have shown more interest than academics in safety, but this is beginning to change.

There are many textbooks on *software testing and quality assurance*, and the topic is also covered in most books on software engineering. Wikipedia is an excellent free resource; start at https://en.wikipedia.org/wiki/Software_testing and follow links relevant to your interests. There is some good work on testing NLP systems, such as Ribeiro et al. [170], but very little explicitly on testing NLG systems. However, some people have found my blog on the topic (<https://ehudreiter.com/2017/02/10/nlg-test-qa/>) to be useful. There are numerous companies that sell tools or consultancy for testing AI systems, but I cannot recommend specific vendors in this book.

Similarly there are many textbooks on *software maintenance*, which is again covered in most books on software engineering, but I am not aware of anything specifically on maintaining NLG systems. It has frequently come up in commercial discussions I have had but does not seem to be discussed in the research literature (we once tried to write a paper on maintenance issues in SumTime; academic reviewers had little interest in the topic). Within the general AI literature, there is work specifically on domain adaptation (e.g. see the survey by Farahani et al. [55]), and I am beginning to see papers on maintaining AI systems more generally, such as [37]. Hopefully more such papers will be published in the future.

Chapter 7

Applications



NLG is being used in many real-world contexts. Indeed, the usage of NLG is evolving extremely rapidly, with new use cases appearing every month. What is possible to accomplish in specific use cases is also changing fast, as better technology opens up new possibilities.

As always with this book, I will focus on fundamentals, not the latest developments. I will first discuss generic issues in applying NLG to real-world use cases. Then I will look at several long-standing use cases of NLG: journalism, business intelligence, summarisation, and healthcare.

7.1 Key Attributes of Successful NLG Applications

7.1.1 *Volume and Scalability*

NLG systems generally need to generate a large number of texts in order to be cost-effective from a commercial perspective; this is related to return on investment (Sect. 5.6.4). For example suppose an NLG system generates a daily summary of the London stock market (on weekdays, not weekends), and that it would take a journalist one hour to write such a summary manually. If the journalist is paid £20 per hour, this means the system saves $£20 \times 52 \times 5 = £5200$ per year. The cost of building and maintaining the NLG is likely to be considerably higher than this! On the other hand, if the system generates 60 stock market summaries each weekday (one summary for each of the 60 largest global stock markets), then it saves £312K per year, which is more likely to be commercially viable.

It is for this reason that successful commercial NLG systems tend to focus on high-volume use cases, such as generating large numbers of local news stories, business intelligence reports, or email summaries. For example in sports reporting, it probably makes more commercial sense to use NLG to produce a large number of

stories about local teams, instead of a small number of stories about world-leading teams.

NLG systems built around prompted models can be considerably cheaper to create than systems built using rules or by training or fine-tuning models. However, if the goal is to create a product (as opposed to an in-house tool that customers do not see), then even if coding costs are massively reduced, there still will be substantial costs for requirements analysis (Chap. 4), evaluation (Chap. 5), testing (Sect. 6.2), and maintenance (Sect. 6.3). Plus of course commercial costs such as marketing. So again volume makes commercial sense.

Of course NLG can be justified in other ways as well, for example the BBC Election Reporter (Sect. 7.2.1) was justified on the basis of speed (need to get lots of election-results stories written quickly) as well as volume. But even here, the system only made sense for generating hundreds of stories about local election results, and it would not have been a viable solution to producing a small number of stories about national results.

A related concept is *scalability*; NLG systems should be usable in many contexts. In healthcare, for example there is limited commercial value in an NLG system which only works in one hospital, such as Babytalk (Sect. 5.6.2). We want systems which can be deployed in many hospitals and hence will work with different systems, sensors, equipment, clinical procedures, and administrative procedures.

Of course, the difficulty of scalability depends on the use case. For example Business Intelligence use cases are often scalable and generalisable at least to some degree, in part because the underlying processes and data are often similar in different companies.

Personal Note

I once worked a research project whose goal was to help non-verbal children create stories about their school days for their parents [24, 198]. We worked with a small number of children and achieved good success with them, but it was clear that our solutions would not work for many other non-verbal children. Children who cannot speak are very different from each other: different levels and types of cognitive impairment, different physical problems, different backgrounds and attitudes, etc. In short we could build bespoke solutions for individual children, but not scalable solutions which worked for a large number of non-verbal children.

7.1.2 Data Availability

We cannot build a data-to-text NLG system unless we have access to the data that the system is describing or summarising; if we are building a neural NLG system

which is trained or fine-tuned, we also need access to training data. Data should also be of high quality. These issues can constrain where commercial NLG makes sense.

For example a limitation on some NLG systems used in sports reporting is that they do not have access to visual data. A human journalist who observes a game will add phrases such as *Jones was in tears after missing a shot* or *Brown jumped for joy after scoring*, and this human-interest content makes stories more interesting to readers. But an NLG system cannot do this if it has access to scores and player performance data but not camera feeds of players' faces.

7.1.3 Accuracy

Neural NLG systems make mistakes (Sect. 4.1.2). Mistakes are more tolerable in some use cases than in others. We can sometimes estimate the number of acceptable mistakes in a use case by analysing mistakes in texts written by people. For example Thomson et al. [197] found an average of 1.5 factual errors in human-written sports stories. So 1.5 mistakes may be acceptable (although of course not desirable) in a sports story; it would not be acceptable in medical reporting.

In many contexts the best way to produce accurate texts is for humans and NLG systems to work together, for example by asking human writers to edit the output of NLG systems (Sect. 4.3.2). If done well, this process can produce texts which are more accurate than texts produced just by humans (because the NLG system will not make human-like errors like copying the wrong number) or just by NLG systems (because the humans will not hallucinate untrue information).

On the other hand, human checking and editing of NLG outputs must be done well in order to be effective. For example CNET published a number of articles produced by an NLG system, which had been checked by human editors, which nevertheless had serious and potentially harmful mistakes, such as giving inaccurate advice about personal finance.¹

Accuracy is especially important if there are safety concerns (Sect. 6.1); Sect. 7.5.5 discusses safety concerns in medical applications of NLG.

7.1.4 Maintainability and Adaptability

As discussed in Sect. 6.3, real-world NLG systems must be maintained and adopted as the domain and user needs change. In healthcare, for example if medical advice is updated due to new research, NLG systems should adapt their output accordingly.

¹ <https://edition.cnn.com/2023/01/25/tech/cnet-ai-tool-news-stories/index.html>

User: I live in Aberdeen and want to see a respiratory health doctor, what should I do?

ChatGPT: If you're in Aberdeen and you're looking to see a respiratory health doctor, here are steps you can take:

1. Contact Your Primary Care Physician:

Start by reaching out to your primary care doctor or general practitioner. They can assess your symptoms, provide initial care, and refer you to a specialist if needed.

2. Insurance Coverage Check:

Check with your health insurance provider to understand your coverage and whether you need a referral to see a specialist.

3. Specialist Referral:

If your primary care doctor determines that you need to see a respiratory health specialist, they can provide you with a referral. This referral may be necessary for insurance purposes.

4. *etc*

Fig. 7.1 ChatGPT advice is not appropriate for Aberdeen, UK, because the UK does not have an insurance-based health system

Unfortunately, this is not always easy to achieve in practice (Sect. 6.1.1.5); Strickland [190] points out that this was a major problem with the IBM Watson health systems.

Of course change is faster in some domains than in others; NLG weather forecast generators, for example, may evolve less quickly than medical NLG systems. Hence, the maintenance burden is different in different applications. Application builders should estimate likely maintenance costs for their system, as well as development costs.

A related problem is that NLG systems should be able to adapt their output to local circumstances. Figure 7.1 shows an example of an adaptation failure; ChatGPT responds to a query with information which makes sense in the USA but not in the UK, since UK patients generally do not have health insurance and do not need to deal with insurance providers. Perhaps ChatGPT thought that *Aberdeen* referred to an American town, but this is odd because Aberdeen, UK, is ten times bigger than any of the American Aberdeen's.

7.1.5 *Acceptability and Trust*

NLG systems will only be used if people trust and accept them. This is partially a *change management* issue, and it is useful to be aware of the literature on managing and enabling organisational change, including introducing new technologies.²

For example it is difficult to get hospitals and doctors to use machine learning models for medical diagnosis; doctors see diagnosis as a core part of their job which they do not want to be automated, and hospitals do not see diagnosis as a major ‘pain point’ which needs to be automated [190]. Paul Meehl [130] showed in 1954 that simple regression models could do some medical diagnosis tasks better than the average doctor; however, Meehl’s models were never used. Strickland analysed the failure of IBM’s Watson health system [190] and highlighted factors such as poor fit to clinical workflows, and lack of understanding of requirements, that is what doctors and hospitals actually wanted help with.

Personal Note

Many years ago, the Scottish AI pioneer Rob Milne (now sadly deceased) told me that the most successful AI applications were those that targeted ‘peripheral’ tasks, i.e. tasks that users saw as annoying distractions. It was much harder to build a successful AI application for a core task, i.e. a task that the user saw as a central aspect of his job and perhaps identity. Diagnosis is a core task for doctors.

With regard to trust, domain experts may have more trust in simple models which they feel they understand, compared to black box neural models. This is not irrational if we consider that black box models reflect their training data, which may be biased and noisy. For example a doctor may be concerned that a neural LLM used in a patient information system may in some cases repeat inappropriate content from the Internet (Sect. 7.5.5). There is no way for a doctor to check the black box LLM to see if this may happen, he needs to trust the system not to do this.

Another issue is that doctors often have additional information (for example visual observation of the patient) which is not available to the NLG system. It is easier for doctors to assess the impact of this information if they understand how the NLG system works.

My experience (and that of other people working in applied NLG) is that trust is easily lost; even a single mistake can lead to users distrusting an NLG system and refusing to use it.

² https://en.wikipedia.org/wiki/Change_management

W 10–15 backing SW by mid afternoon and S 13–18 by midnight.

Fig. 7.2 Extract from SumTime forecast, in appropriate sublanguage

7.1.6 Conforming to Genre and Sublanguage

In some applications, texts must be written in a way which conforms to a *sublanguage* [94]. That is they need to use words and syntactic structures which are expected in the target domain and genre.

For example the SumTime system generated marine weather forecasts [167]. An example is shown in Fig. 7.2. Note that this sentence is *not* a grammatically correct sentence according to the normal rules of English grammar; for example it does not have a correctly inflected clausal verb. It does however fit the rules of the marine-forecast sublanguage, so it is perfectly acceptable.

Similarly, texts intended for clinicians and medical professionals should use appropriate medical terminology and language. Indeed, Moramarco et al.'s [138] evaluation of Note Generator (Sect. 7.5.1.1) identified several cases where inappropriate acronyms were used, and classified these as errors.

Precision terminology is also very important in some cases. For example *bad headache* and *migraine* are sometimes used interchangeably by patients, but to doctors they have different meanings.

Personal Note

Patients may want to see some medical terminology, even if they do not fully understand it. For example in our Babytalk system for parents (Sect. 2.2.2), we initially tried to remove medical terminology. However, some parents did not like this and said that we were ‘dumbing down’ texts and patronising them, so we put some medical terminology back into the parent reports, even though in some cases this made them harder to understand.

7.2 Journalism

One of the oldest use cases for NLG is journalism and media, that is using NLG to produce stories for newspapers and websites. Typically, data-to-text NLG is used for *data journalism* where an article is largely based on data of some kind; for example financial stories that summarise financial data, or sports stories that summarise what happened in a sports match based on match data. This is sometimes called *automatic journalism* or *robo-journalism*.

Florence Eshalomi has been elected MP for Vauxhall, meaning that the Labour Party holds the seat with a decreased majority.

The new MP beat Liberal Democrat Sarah Lewis by 19,612 votes. This was fewer than Kate Hoey's 20,250-vote majority in the 2017 general election.

Sarah Bool of the Conservative Party came third and the Green Party's Jacqueline Bond came fourth.

Voter turnout was down by 3.5 percentage points since the last general election.

More than 56,000 people, 63.5% of those eligible to vote, went to polling stations across the area on Thursday, in the first December general election since 1923.

Three of the six candidates, Jacqueline Bond (Green), Andrew McGuinness (The Brexit Party) and Salah Faissal (independent) lost their £500 deposits after failing to win 5% of the vote.

This story about Vauxhall was created using some automation.

Fig. 7.3 Example story from BBC election reporter, from <https://www.bbc.co.uk/news/technology-50779761>. This is the same as Fig. 1.6

Diakopoulos [48] presents a detailed summary of 'Automated Content Production' in journalism up to 2019. In this period, the focus was on using relatively simple rule-based NLG techniques (sometimes just templates) to generate drafts of relatively short and straightforward articles; these articles were post-edited by journalists (Sect. 4.3.2) before they were released.

Large language model technology (which appeared after 2019) allows more complex articles to be generated. At the time of writing, there is a lot of excitement about using ChatGPT and other large language models in journalism; they should expand the power of automatic journalism by semi-automating production of more types of stories. Perhaps the technology will also enable the development of new types of news media, such as dialogue systems for news (where users can ask questions about news stories).

7.2.1 Example: BBC Election Reporter

A perhaps typical example of using relatively simple rule-based NLG techniques to produce media articles is the BBC's election reporter system, which generated short reports about election results in individual constituencies in the 2019 UK general election.³ An example is shown in Fig. 7.3. Stories for constituencies in Wales were produced in Welsh as well as in English. Stories were checked by human journalists before they were released; in a few cases the journalists added additional content, generally deeper analysis of results in key races.

The rationale for this project is that the BBC can report election results overall, but it cannot manually produce 650 stories about individual MP races in the required

³ <https://www.bbc.co.uk/news/technology-50779761>

time frame (readers expect to see these stories the morning after the election, not several days later). The focus on fast production of a large number of news stories, with human oversight and editing, is common in automatic journalism.

This project used Arria NLG Studio, which is a commercial tool for building rule-based NLG. Implementation details are not publicly available, but interested readers may wish to look at Valteri [111, 131], which is an election reporting system built by an academic research group that also used rule-based NLG.

7.2.2 *Types of News*

7.2.2.1 General News

A number of NLG systems have been used to produce general news articles [48], using an approach similar, at least at a conceptual level, to the BBC election reporter (Sect. 7.2.1). For example Radar⁴ used Arria NLG Studio to generate a large number of localised articles from data sets about politics, crime, health, traffic, and so forth. For instance, the Radar system would ingest a UK-wide dataset about obesity rates, extract the relevant information for a particular town or city (such as Aberdeen), and then generate a story based on this data (i.e. obesity in Aberdeen). This story would be sold to a local newspaper, which would release it, possibly after being edited by a journalist to add more local content and context.

The advantage of this approach for journalists is that it made it much easier for local newspapers, which often were short-staffed and lacked journalists with domain expertise, to publish data-based articles on health, crime, etc. A large national newspaper such as *New York Times* or *Guardian* will have in-house teams of specialist data journalists, but small local newspapers cannot afford this.

Large language models such as ChatGPT may have a large impact in this area; technology and use cases are evolving very rapidly at the time of writing.

7.2.2.2 Sports Reporting

One of the most active areas of NLG in media is sports reporting. Modern sporting events are often monitored by sensors which produce large amounts of data; there is also often a large amount of historical data about the players and teams involved in the match. This data can be used by NLG systems to generate many kinds of articles, including live updates during a match, post-match game descriptions, and pre-match summaries of the teams involved. Articles can be targeted towards different audiences, such as the general public, fans of one of the teams, or people placing bets; they can be delivered on social media, websites, and indeed printed

⁴ <https://pa.media/radar/>

stories. NLG technology can also be used to generate reports from sports data for coaches and talent scouts.

Sports stories are an area where *variation* (Sect. 4.2.2) is very important. Sports fans read lots of sports stories, and it is essential that these stories use different words and sentences (even when conveying similar information); otherwise readers will get bored and may stop reading the stories. Certainly in the sports NLG projects I have been involved in, considerable effort was put into supporting extensive variation.

Anyways, the richness of data and use cases makes this an exciting area for NLG. It is also an area where there is a lot of interest in using neural NLG techniques. In part this is because from a quality-factor perspective (Sect. 4.1), it is essential that sports stories be well written and engaging. Errors (such as hallucination) are of course not desirable but may have less impact than in safety-critical contexts such as clinical decision support. An error in a story about a children's basketball game is not going to kill anyone!

7.2.2.3 Financial News

There is also a lot of interest in using NLG in financial journalism. Investors and business people more generally are very interested in knowing what is happening in companies of interest to them. There are thousands of companies listed on the London Stock Exchange, New York Stock Exchange, NASDAQ, etc., and it is difficult for human journalists to write stories about business developments in such a large number of companies. But since basic data about listed companies is available from the stock markets themselves and also from relevant regulators, NLG technology can be used to generate large numbers of stories about these companies, as well as stories about stock market performance.

Indeed, the very first work on data-to-text NLG, by Karen Kukich in 1983 [103], was on generating summaries of stock market activity. A simple example from Kukich's system is shown in Fig. 7.4. Of course modern financial reporting NLG systems generate much more sophisticated texts about a wide range of financial stories (not just stock market updates), but the core concept is the same; the system takes financial data as input and produces a story about this data as output.

As with sports, a key issue is data availability. Generated stories are richer when the NLG system has access to more data about companies and markets,

After climbing steadily through most of the morning, the stock market was pushed downhill late in the day. Stock prices posted a small loss, with the indexes turning in a mixed showing yesterday in brisk trading.

The Dow Jones average of 30 industrials surrendered a 16.28 gain at 4pm and declined slightly , finishing the day at 1083.61, off 0.18 points.

Fig. 7.4 Stock market story from world's first data-to-text system [103]

perhaps including data extracted from free-text documents using natural language understanding techniques.

From a quality-factor perspective, financial stories should of course to be well written and varied, like all news stories. They also must be accurate, because investors may make decisions based in part on news stories. Accuracy in financial news is perhaps not as important as in medical decision support, but it is considerably more important than accuracy in sports reporting.

From a technology perspective, at the time of writing many financial news generators use rule-based techniques, but there is huge interest in generating financial stories using machine learning and neural techniques. However, it is worth noting that Thomson-Reuters generated financial stories for a number of years using a very interesting machine learning approach (essentially ML techniques were used to learn and select templates [101]), but then withdrew this system, I suspect in part because journalists wanted more explicit control over the content of generated stories. In short, this particular ML NLG system may not have been acceptable to journalists in real-world usage (Sect. 7.1.5). Of course other ML techniques may be more successful, but acceptability can only be assessed when the system is in production real-world usage by many journalists.

7.2.3 *Fake News*

Neural NLG systems make mistakes. Reputable journalists and media organisations will of course do their best to detect and minimise mistakes. However, there are also malicious and unscrupulous agents who deliberately use NLG to produce stories which are not true. Large language models unfortunately can be very good at producing ‘fake news’ [185].

A lot of fake news is simply designed to attract readers and hence get money from advertisers. Of course some media outlets have printed fake stories about alien abductions and the like for decades, without any use of AI and NLG! But creating plausible-sounding stories about visits from little green men is easier and quicker with NLG.

More damaging is using AI to create fake material which people pay money to read. For example at the time of writing there is a lot of concern about people buying useless AI-written travel guides.⁵

Perhaps most damaging is fake news which tries to influence people. In the last three months of the 2016 US Presidential election, fake news got more engagement in Facebook than genuine news.⁶ These articles were presumably written by humans, not NLG systems. It is not clear to what degree NLG systems are currently

⁵ <https://www.nytimes.com/2023/08/05/travel/amazon-guidebooks-artificial-intelligence.html>

⁶ <https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook>

being used to generate fake news for such purposes, but this is a major safety concern at the societal level [178].

7.3 Business Intelligence

Another long-standing use case of NLG is Business Intelligence (BI).⁷ BI tools help people understand data about their organisation (business, charity, government, etc.). Amongst other things, this includes data about:

- Sales, profits, losses
- Employees
- Budgets
- Inventories and logistics
- etc.

For example a BI tool could help an organisation understand that most of its sales growth is in Asia, turnover in IT staff is growing rapidly, its inventory of laptops is too large, etc. This in turn helps managers and other decision-makers make good decisions.

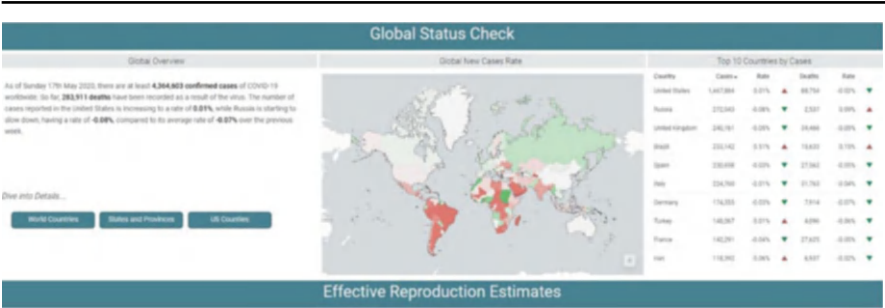
Most commercial BI tools, such as Tableau and PowerBI, focus on graphical presentation of data. However, in some cases textual presentation of data is better (Sect. 4.4), not least because it can include analyses and background information to supplement the data.

From a commercial perspective, one of the big advantages of BI as a use case is that it is often scalable (Sect. 7.1.1). Whereas medical NLG systems often need to be tailored for specific hospitals, and journalism NLG systems need to be adapted for specific local markets, all organisations have budgets and employees, and most have sales and inventories. Furthermore, the data about budgets, employees, sales, and inventories is often held in databases or spreadsheets, many of which have a similar structure in part because of the requirements of tax and regulatory authorities. Hence it is possible to build somewhat generic NLG BI systems which can be sold to many customers.

7.3.1 Example: Covid Reporter

A simple example is shown in Fig. 7.5. This system was created by Tibco and Arria to provide information to the public about the status of the Covid-19 pandemic, using business intelligence techniques. The full system includes many information displays; the one shown in Fig. 7.5 gives summary information about

⁷ https://en.wikipedia.org/wiki/Business_intelligence



NLG text in above screenshot (replicated for clarity):
As of Sunday 17th May 2020, there are at least **4,364,603 confirmed cases** of COVID-19 worldwide. So far, **283,911 deaths** have been recorded as a result of the virus. The number of cases reported in the United States is increasing to a rate of **0.01%**, while Russia is starting to slow down, having a rate of **-0.08%**, compared to its average rate of **-0.07%** over the previous week.

Fig. 7.5 Business Intelligence about Covid-19. Taken from <https://www.tibco.com/covid19>. This page no longer exists, it was removed when the pandemic ended

the presence of Covid-19. It uses an NLG-generated texts to give a high-level summary, a map to show distribution of Covid-19 in different countries, and a table to give more detailed information about badly affected countries. This ‘multi-modal’ combination of text and graphics is typical and allows each media to be used to communicate the type of information it is best suited for.

7.4 Summarisation

Another long-standing use of NLG is summarising documents. Indeed, summarisation predates NLG, with some initial work in this area done in the 1950s [117] focusing on statistical techniques for identifying key material in a document.

There are two general approaches to summarisation. *Extractive summarisation* identifies key sentences and builds a summary from these, while *abstractive summarisation* creates new sentences to summarise information. A simple example is shown in Fig. 7.6, which shows possible extractive and abstractive summaries of an Aberdeen weather forecast. The extractive one is simply the first sentence in the text, while the abstractive summary integrates information from several source sentences into the summary sentence.

Another key distinction is between *single-document summaries* (where one document is summarised, as in the example in Fig. 7.6) and *multiple-document summaries* (where multiple documents are summarised, for example a summary of a set of news articles). The latter are often more useful, especially in contexts (such as summarising reviews or email chains) when the reader wants to quickly get an understanding of a set of documents.

Source text (weather forecast for Aberdeen):

A mix of brighter spells and showers. Best of the brightness in the morning, particularly across Moray and western Aberdeenshire. Showers isolated early before building through the afternoon. Winds light. Maximum temperature 8 °C.

Extractive summary:

A mix of brighter spells and showers.

Abstractive summary:

A mild day with a mix of brighter spells (especially in the morning) and showers.

Fig. 7.6 Extractive and abstractive summaries of a weather forecast

There are many uses of summarisation, including:

- *Meeting summaries:* The Note Generator system (Sect. 7.5.1.1) is an example of a system that generates summaries of meetings. There are many other tools in this space, which generate different types of summaries (key points, action items, etc.) of meetings. Rennard et al. [169] survey research in this area.
- *Email summaries:* Likewise there are many tools which summarise email messages (individually or collectively), in order to help people manage overflowing inboxes; for example Microsoft Copilot in Outlook summarises email threads.
- *News summaries:* Extensive research has been done on summarising news articles, initially driven by the desire of intelligence agencies to help analysts quickly identify key information [123]. Recent work in this area has also looked at summarising social media.
- *Review summaries:* There is also a lot of interest in summarising product reviews, and in this case it is essential that the summary covers a set of reviews (a summary of one review is less useful). Amazon provides summaries of reviews in its ecommerce site at the time of writing.

At the time of writing, summarisation is usually done using large language models.

7.4.1 Example: Summarising Emails with Google Bard

Google Bard is one of many systems that can summarise emails, including chains and sets of emails as well as single emails. I show a simple Bard example in Fig. 7.7. In this case, Bard has found the relevant email (which is pretty straightforward), extracted key information from the email which is relevant to my question, and created a sentence which communicates this information. This is an example of generating an abstractive summary of a single document.

Question to Bard:

Check my emails, what is the next book in my bookclub

Response from Bard:

The next book for the AJSCC book club is *Odessa Stories* by Isaac Babel. The meeting will be on Thursday, January 18th, 2024 at 8pm on Zoom.

Source email:

Subject: dates of next few meetings for AJSCC book club

We had a wonderful evening of discussion around Susan's book choice! Lovely to have so many of us participating. Here are full details of our next two books for January and March

date: Thursday 18th January 2024

time: 8.00pm

location: Zoom

book: *Odessa Stories*

author: Isaac Babel

publisher: Pushkin Collection

published: 2016

ISBN: 978-1782271666

chosen by: Ehud

additional email content not shown

Fig. 7.7 Example of Bard summarising an email

7.5 Medical Applications

For most of my career, I have been very interested in applications of NLG to healthcare. In principle, we can use NLG in all sorts of ways to help clinicians and patients, including automated reporting (save doctor's time and perhaps reduce errors), clinical decision support (help clinicians make good decisions and avoid mistakes), patient information (keep patients informed and help them make appropriate decisions), and behaviour change (encourage patients to adopt healthier behaviours). Success in this area would have huge impact, both financially (according to WHO, healthcare is over 10% of the world's economy) and (more importantly) on people's lives. Indeed, when I was asked to write a long-term vision for my research in the early 2000s, I said that I wanted to use NLG to help patients live healthier lives.

However, healthcare is also a very challenging area to work in, especially if our goal is real-world impact instead of academic papers. In the early 1990s, a colleague told me that the challenge with AI generally in healthcare was building scalable robust solutions which doctors and health organisations would adopt. Building a system that works well on one data set or in one hospital is much easier than building a system which can be widely used; and building a widely usable system is much

easier than getting health organisations and professions to actually use the system (Sect. 7.1.5).

In medicine, it is also the case that mistakes can hurt or even kill people; for this reason, we need to be careful when using neural language models. Indeed, the terms of use of the BLOOM language model explicitly prohibit using BLOOM ‘to provide medical advice and medical results interpretation’.⁸

In short, NLG applications in healthcare have huge potential, but we must also keep in mind that it is difficult to deploy NLG technology on a large scale in healthcare, even if the technology is safe and effective.

7.5.1 *Use Case: Reporting*

Many clinicians are most excited by NLG systems that automate or otherwise support reporting tasks. Doctors are expected to write large numbers of reports and documents, and this takes up a considerable amount of their time. Most doctors see this as a peripheral task (Sect. 7.1.5); it needs to be done, but they welcome innovations which reduce the amount of time they spend on writing documents. Diagnosis, in contrast, is a core task, where automation is resisted; Strickland [190] discusses a related point. Hence, from a change management perspective (Sect. 7.1.5), many clinicians welcome AI support for reporting, but not for diagnosis.

In other words, doctors will probably be receptive to ‘We will use AI to automate paperwork so that you have more time for careful decision-making’. However, they may not be receptive to ‘We will use AI to automate decision-making so that you have more time to carefully complete your paperwork’.

From a commercial evaluation perspective, the benefits of report automation (time saved by doctors) are also relatively straightforward to measure. Changes in patient outcomes can be harder to measure; for example if we are interested in survival rate 5 years after treatment, we will have to wait five years to get this information.

From a scalability perspective (Sect. 7.1.1), reporting practices differ in different healthcare organisations, which means that reporting systems are often specific to individual hospitals or even hospital units. An exception is regulatory reporting, which of course is standardised to the requirements of the regulator; for example I have seen commercial work on using NLG to generate clinical safety reports (on pharmaceuticals) for regulatory agencies.

⁸ <https://huggingface.co/spaces/bigscience/license>

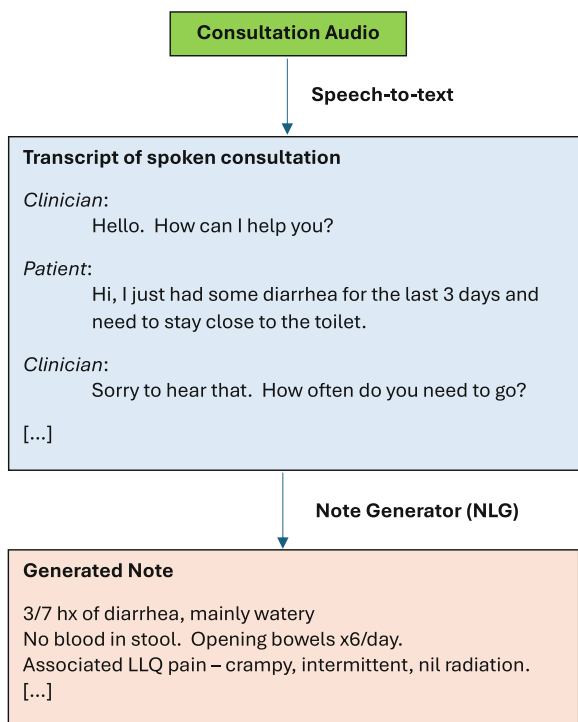


Fig. 7.8 Example of Note Generator (based on [95]); this is the same as Fig. 1.3

7.5.1.1 Example: Summarising Consultations

An example of an NLG system used for medical reporting is Note Generator (described in Sect. 1.3). It generates a summary of a doctor–patient consultation, which is entered into the patient’s medical record after being checked and edited by a doctor. An extract from Note Generator was shown in Fig. 1.3; I repeat this in Fig. 7.8 for convenience.

Manually writing a summary of a doctor–patient consultation can take 2–3 minutes, which is a sizeable chunk of the ten minutes allocated to consultations with general practitioners (GPs) in the UK. Partially automating this task gives doctors more time to interact with patients, which they welcome. Moramarco [137] analysed real-world usage of Note Generator (Sect. 5.5.2) and reported that doctor using Note Generator (i.e. post-editing a Note Generator summary) could generate a summary in 9% less time compared to manually writing summaries from scratch. Summaries produced by post-editing Note Generator drafts were also slightly more accurate than manually written summaries.

7.5.2 Use Case: Patient Information and Behaviour Change

It is a truism in medicine that the best way to improve most people's health is to change their behaviour. This includes adopting healthier life styles (diet, exercise, not smoking, etc.), correctly complying with medical instructions (e.g. take medication correctly), and contacting healthcare personnel when appropriate. Indeed, it is hard to see how healthcare systems can cope with increasingly elderly populations unless people look after themselves better.

Also, modern medicine expects patients to make informed decisions about their healthcare, and informed decisions are only possible if patients understand benefits and risks. This is difficult for many people, especially if probabilities are involved; many people have a weak understanding of risk and probabilities (Sect. 7.5.2.2).

Hence there is a lot of potential in building NLG systems that directly interact with patients in order to educate people about health consequences of their behaviour, encourage behaviour change, and/or explain important information. I personally think that this is an area of medicine where NLG could potentially make very important contributions.

A general rule is that we can help and encourage people to do things they would like to do, but it is very hard to change intentions. For example looking at smoking cessation, we can give suggestions to people who want to stop smoking about how to achieve this goal; it is much harder to try to convince committed smokers to stop smoking. Similarly, we can support people who want to make informed decisions by giving them appropriate information, but giving information is not going to do much good if the patient wants to delegate all decisions to his or her doctor.

Behaviour change is difficult even for people who want to change, especially on a long-term basis. For example most adult smokers in the UK want to stop smoking, but quitting is still very hard. Similarly most overweight and obese people would like to lose weight and may achieve this temporarily (e.g. by following a diet), but find it harder to lose weight on a sustained long-term basis.

In part for the above reasons, it is very useful for NLG systems (and indeed AI systems more generally) in this space to be integrated into the broader healthcare system, so that they are just one of many tools which people can use to help them change their behaviour. Unfortunately, differences between healthcare systems can make this difficult.

7.5.2.1 Example: Encouraging Smoking Cessation

The STOP system (described in Sect. 5.1) generated personalised smoking-cessation advice for smokers, based on their answers to a smoking questionnaire. An extract from STOP was shown in Fig. 5.2; I repeat this in Fig. 7.9 for convenience.

Unfortunately, a randomised controlled clinical trial showed that the STOP system was not effective, i.e. people who received a fixed letter about smoking

Dear Ms Cameron

Thank you for taking the trouble to return the smoking questionnaire that we sent you. It appears from your answers that although you're not planning to stop smoking in the near future, you would like to stop if it was easy. You think it would be difficult to stop because *smoking helps you cope with stress, it is something to do when you are bored, and smoking stops you putting on weight*. However, you have reasons to be confident of success if you did try to stop, and there are ways of coping with the difficulties.

Fig. 7.9 Part of a STOP leaflet; this is the same as in Fig. 5.2

cessation (not produced by an NLG system) were at least as likely to stop smoking as people who received a letter produced by STOP [110, 165].

7.5.2.2 Communicating Risks and Probabilities

It is often necessary in patient information systems to communicate risks and probabilities. Unfortunately, many patients in the UK, USA, and elsewhere have limited numeracy skills and may not understand numbers, so risk should be communicated to them in other ways, such as with words.

Some of the challenges in communicating risk linguistically are described by Berry [20–22]; see also Hommes et al. [80]. For example:

- Different people interpret descriptors such as *likely* and *common* in different ways.
- It matters how risk is framed (risk of success vs. risk of failure).
- Is risk communicated in absolute or relative terms (*this procedure has a 5% chance of success* vs. *this procedure has a 50% higher chance of success than the alternative*)?

Even more complexity arises if we wish to communicate the reliability of a risk, e.g. *the model says you have a 10% chance of developing lung cancer; however, the risk may be higher because the model has ignored the fact that both of your parents died of lung cancer* [183].

Unfortunately, while there is a large general literature on risk communication, I am not aware of much research in the NLG community on communicating risk in a way which takes these issues into account.

7.5.3 Use Case: Clinical Decision Support

NLG systems can summarise information and help doctors make decisions. This is an appealing concept in principle, but success is difficult, in part because of the issues mentioned in Sect. 7.5.1. Doctors enjoy decision-making and generally

Example BT45 output (extract):

By 11:00 the baby had been hand-bagged a number of times causing 2 successive bradycardias. She was successfully re-intubated after 2 attempts. The baby was sucked out twice. At 11:02 FIO2 was raised to 79%.

Fig. 7.10 Example outputs from Babytalk BT45 decision support system; this is the same as shown in Fig. 2.3

are pretty good at it; they also have access to additional information sources not available to the NLG system (e.g. observing and talking to the patient). So help is often neither wanted nor needed [190].

Safety issues (Sect. 6.1) are very important in this area; mistakes could cost lives and also could lead to massive lawsuits. Trust is paramount; if doctors don't trust an NLG system (and even a single mistake can destroy trust), they will not use it. Adding explanations, if done well, could enhance trust.

Personal Note

Many years ago I was very excited by using NLG as a decision support tool, by summarising key information for clinicians. Looking back at this now, I think this may be possible technically, but getting clinicians to use such a system would be very hard. Adoption will probably be significantly easier for systems focusing on the other use cases discussed here (reporting, behaviour change, business intelligence).

7.5.3.1 Example: ICU Decision Support

Babytalk (described in Sect. 2.2.2) generated decision support material, nursing shift handover reports, and parent reports for babies in a neonatal intensive care unit, using information extracted from the baby's electronic patient record. An extract from the Babytalk systems was shown in Fig. 2.3; I repeat the extract from the Babytalk decision support system in Fig. 7.10 for convenience.

7.5.4 Medical Business Intelligence Use Cases

Of course medicine is a huge area, and there are many other ways we can use NLG in healthcare. In particular, NLG can be used in many healthcare contexts to provide business intelligence (BI) (Sect. 7.3) which supports non-clinical processes such as logistics and quality control.

For example we can use NLG BI to support logistics and help insure that equipment, medication, blood supplies, ambulances, etc., are in the right place. I am not aware of research projects in this space, but I have seen some commercial uses of NLG that support BI in healthcare. A medical colleague once told me that she thought logistics was the best place to apply AI in healthcare, since it is extremely important, and generally raises fewer concerns about safety, acceptability to clinicians, etc.

Another use of NLG+BI is to provide quality-of-service information on healthcare, either to patients (summarising quality and outcome data about a hospital) or to doctors (summarising how well they are doing at reaching various quality targets). Management can also use NLG+BI to analyse efficiency and outcome data. Again I am not aware of research projects in this area, and it seems mostly of interest to the commercial NLG world.

Last but not least, we can use NLG+BI to summarise and explain public health data, especially in local areas where public health officials may lack data analysis expertise. I have seen discussions about this but not actual projects, which is a shame.

7.5.5 *Safety*

Safety (Sect. 6.1) is obviously very important in medical applications, especially since incorrect medical information or advice could injure or kill people. Hallucinations are unacceptable in texts that impact medical care, as are omissions of key information. Singhai et al. [182] present a formal evaluation of a medical language model, MedPaLM; Tamayo-Sarver [193] describes some of the failings he has encountered when trying to apply ChatGPT to real-world medical situations. Readers should keep in mind that humans as well as AI systems make mistakes, and a workflow which combines human and AI can make fewer mistakes (Sect. 4.3.2) and hence perhaps be safer, than a workflow that uses just humans (or just AI systems).

When systems are interacting with patients, it is also essential to minimise adverse emotional effects. Of course discussing medical details and options can be inherently depressing for patients [209], but systems should not unnecessarily make people feel depressed, inadequate, etc. As discussed in Sect. 6.1.1.4, Balloccu [10] asked domain experts to analyse texts produced by ChatGPT to response to dietary struggles, and he found many cases responses were factually correct but inappropriate and indeed unsafe because of their potential emotional impact.

Balloccu's domain experts also expressed a general concern that ChatGPT seemed to largely be trained on texts from Internet forums which were not written by healthcare professionals.

Other medical professionals have made similar comments to me about the outputs of large language models; the text was correct but not an appropriate thing to say to a patient. In general, they feel that health information should come from

reputable medical sources, and are concerned that Internet-trained LLMs may rely on sources that are not quality-controlled, such as Reddit forums. Certainly in my own personal experience, advice from well-meaning lay people such as friends and family (as opposed to advice from health professionals) has often been insensitive to emotional issues.

7.6 Further Reading

The usage of NLG in real-world applications is evolving and changing very rapidly. There are numerous commercial websites which proclaim the wonders of specific NLG products, as well as numerous consulting firms who offer advice and analysis on NLG applications. This material is usually very well written and presented but of course is intended to further the author's commercial objectives. It also tends to emphasise case studies, demos, and stories instead of rigorous evaluation.

Diakopoulos et al. [49] is an excellent survey of how large language models and NLG are being used by journalists in 2024. While this is about journalism, many of the insights about use cases, workflows, ethics, etc., are generic and apply to other applications as well; I highly recommend this survey to anyone developing NLG applications. Diakopoulos also wrote a book in 2019 on 'Automating the newsroom' [48] and helps run a website on NLG in journalism (<https://generative-ai-newsroom.com/>); both of these are also excellent sources.

Anyone interested in using NLG or indeed any form of AI in healthcare should read Strickland's excellent retrospective on why IBM's Watson system was not successful in healthcare [190]; I would love to see more such retrospectives! Hüske-Kraus's 2003 survey of NLG in clinical medicine [84] is still useful; technology has changed radically since then, but many of the broader issues have not.

Dale occasionally writes insightful articles about commercial NLG, such as [40].

My personal blog includes many blogs about applications of NLG and the issues raised in this chapter, including:

- *Change management*: <https://ehudreiter.com/2020/01/27/ai-professionals-also-focus-on-change-management/>
- *BBC Election Reporter* (Sect. 7.2.1: <https://ehudreiter.com/2019/12/23/election-results-lessons-from-a-real-world-nlg-system/>)
- *Challenges in medical NLG*: <https://ehudreiter.com/2021/06/21/pain-points-in-health-nlg/>
- *Covid reporter* (Sect. 7.3.1): <https://ehudreiter.com/2020/05/21/adding-narrative-to-a-covid-dashboard/>
- *Sports NLG*: <https://ehudreiter.com/2022/03/21/sports-nlg-commercial-vs-academic-perspective/>

I have written many other blogs about NLG applications which readers may find useful.

References

1. Latin Square Designs: The Concise Encyclopedia of Statistics. Springer, New York, NY (2008). https://doi.org/10.1007/978-0-387-32833-1_223
2. Abed, W., Reiter, E.: Arabic NLG language functions. In: Proceedings of the 13th International Conference on Natural Language Generation, pp. 7–14. Association for Computational Linguistics, Dublin (2020). <https://aclanthology.org/2020.inlg-1.2>
3. Abercrombie, G., Cercas Curry, A., Dinkar, T., Rieser, V., Talat, Z.: Mirages. on anthropomorphism in dialogue systems. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 4776–4790. Association for Computational Linguistics, Singapore (2023). <https://doi.org/10.18653/v1/2023.emnlp-main.290>. <https://aclanthology.org/2023.emnlp-main.290>
4. Abercrombie, G., Rieser, V.: Risk-graded safety for handling medical queries in conversational AI. In: Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 234–243. Association for Computational Linguistics, Online only (2022). <https://aclanthology.org/2022.aacl-short.30>
5. Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., et al.: Guidelines for human-AI interaction. In: Proceedings of the 2019 Chi Conference on Human Factors in Computing Systems, pp. 1–13 (2019)
6. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete Problems in AI Safety (2016). <https://arxiv.org/abs/1606.06565>
7. Appelt, D.E.: Planning English referring expressions. *Artif. Intell.* **26**(1), 1–33 (1985). [https://doi.org/10.1016/0004-3702\(85\)90011-6](https://doi.org/10.1016/0004-3702(85)90011-6). <https://www.sciencedirect.com/science/article/pii/0004370285900116>
8. Arun, A., Batra, S., Bhardwaj, V., Challa, A., Donmez, P., Heidari, P., Inan, H., Jain, S., Kumar, A., Mei, S., Mohan, K., White, M.: Best practices for data-efficient modeling in NLG: how to train production-ready neural models with less data. In: Proceedings of the 28th International Conference on Computational Linguistics: Industry Track, pp. 64–77. International Committee on Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.coling-industry.7>. <https://aclanthology.org/2020.coling-industry.7>
9. Balakrishnan, A., Rao, J., Upasani, K., White, M., Subba, R.: Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 831–844. Association for Computational Linguistics, Florence (2019). <https://doi.org/10.18653/v1/P19-1080>. <https://aclanthology.org/P19-1080>

10. Balloccu, S.: Emotion aware and tailored diet coaching with natural language generation. Ph.D. thesis, University of Aberdeen (2023)
11. Balloccu, S., Reiter, E.: Comparing informativeness of an NLG chatbot vs graphical app in diet-information domain. In: Proceedings of the 15th International Conference on Natural Language Generation, pp. 156–185. Association for Computational Linguistics, Waterville and virtual meeting (2022). <https://aclanthology.org/2022.inlg-main.13>
12. Balloccu, S., Reiter, E., Li, KJH., Sargsyan, R., Kumar, V., Reforgatio, R., Riboni, R., Dusek, O. Ask the experts: sourcing a high-quality nutrition counseling dataset through Human-AI collaboration. In: Findings of The 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2024)
13. Balloccu, S., Schmidtová, P., Lango, M., Dusek, O.: Leak, cheat, repeat: data contamination and evaluation malpractices in closed-source LLMs. In: Graham, Y., Purver, M. (eds.) Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 67–93. Association for Computational Linguistics, St. Julian's (2024). <https://aclanthology.org/2024.eacl-long.5>
14. BBC: How a chatbot encouraged a man who wanted to kill the queen (2023). <https://www.bbc.co.uk/news/technology-67012224>
15. Belz, A., Agarwal, S., Shimorina, A., Reiter, E.: A systematic review of reproducibility research in natural language processing. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 381–393. Association for Computational Linguistics, Online (2021). <https://doi.org/10.18653/v1/2021.eacl-main.29> <https://aclanthology.org/2021.eacl-main.29>
16. Belz, A., Mille, S., Howcroft, D.M.: Disentangling the properties of human evaluation methods: a classification system to support comparability, meta-evaluation and reproducibility testing. In: Proceedings of the 13th International Conference on Natural Language Generation, pp. 183–194. Association for Computational Linguistics, Dublin (2020). <https://aclanthology.org/2020.inlg-1.24>
17. Belz, A., Thomson, C.: The 2023 RepronLP shared task on reproducibility of evaluations in NLP: overview and result. In: Proceedings of the 2023 Workshop on Human Evaluation (2023). <https://aclanthology.org/2023.humeval-1.4/>
18. Belz, A., Thomson, C., Reiter, E.: Missing information, unresponsive authors, experimental flaws: the impossibility of assessing the reproducibility of previous human evaluations in NLP. In: The Fourth Workshop on Insights from Negative Results in NLP, pp. 1–10. Association for Computational Linguistics, Dubrovnik (2023). <https://aclanthology.org/2023.insights-1.1>
19. Bender, E.M., Friedman, B.: Data statements for natural language processing: toward mitigating system bias and enabling better science. *Trans. Assoc. Comput. Linguist.* **6**, 587–604 (2018). https://doi.org/10.1162/tacl_a_00041
20. Berry, D.: EBOOK: Risk, Communication and Health Psychology. McGraw-Hill Education, New York (2004)
21. Berry, D.: Health Communication: Theory and Practice: Theory and Practice. McGraw-Hill Education, New York (2006)
22. Berry, D.C., Knapp, P.R., Raynor, T.: Is 15 per cent very common? Informing people about the risks of medication side effects. *Int. J. Pharm. Pract.* **10**(3), 145–151 (2011). <https://doi.org/10.1111/j.2042-7174.2002.tb00602.x>
23. Bickmore, T.W., Trinh, H., Olafsson, S., O'Leary, T.K., Asadi, R., Rickles, N.M., Cruz, R.: Patient and consumer safety risks when using conversational assistants for medical information: an observational study of Siri, Alexa, and Google assistant. *J. Med. Internet Res.* **20**(9), e11510 (2018). <https://doi.org/10.2196/11510> <http://www.jmir.org/2018/9/e11510/>
24. Black, R., Waller, A., Turner, R., Reiter, E.: Supporting personal narrative for children with complex communication needs. *ACM Trans. Comput.-Hum. Interact.* **19**(2) (2012). <https://doi.org/10.1145/2240156.2240163>
25. Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia,

- L., Turchi, M.: Findings of the 2017 conference on machine translation (WMT17). In: Proceedings of the Second Conference on Machine Translation, pp. 169–214. Association for Computational Linguistics, Copenhagen (2017). <https://doi.org/10.18653/v1/W17-4717>. <https://aclanthology.org/W17-4717>
26. Braun, D., Klimt, K., Schneider, D., Matthes, F.: SimpleNLG-DE: adapting SimpleNLG 4 to German. In: Proceedings of the 12th International Conference on Natural Language Generation, pp. 415–420. Association for Computational Linguistics, Tokyo (2019). <https://doi.org/10.18653/v1/W19-8651>. <https://aclanthology.org/W19-8651>
 27. Braun, D., Matthes, F.: AGB-DE: a corpus for the automated legal assessment of clauses in German consumer contracts (2024). <https://arxiv.org/abs/2406.06809>
 28. Braun, D., Reiter, E., Siddharthan, A.: Saferdrive: an NLG-based behaviour change support system for drivers. *Nat. Lang. Eng.* **24**(4), 551–588 (2018). <https://doi.org/10.1017/S1351324918000050>
 29. Braun, V., Clarke, V.: Thematic analysis. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), *APA handbook of research methods in psychology*, Vol. 2. Research designs: Quantitative, qualitative, neuropsychological, and biological (pp. 57–71). American Psychological Association (2012). <https://doi.org/10.1037/13620-004>
 30. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020)
 31. do Carmo, F., Shterionov, D., Moorkens, J., Wagner, J., Hossari, M., Paquin, E., Schmidtke, D., Groves, D., Way, A.: A review of the state-of-the-art in automatic post-editing. *Mach. Transl.* **35**, 101–143 (2021)
 32. Cascallar-Fuentes, A., Ramos-Soto, A., Bugarín Diz, A.: Adapting SimpleNLG to Galician language. In: Proceedings of the 11th International Conference on Natural Language Generation, pp. 67–72. Association for Computational Linguistics, Tilburg University (2018). <https://doi.org/10.18653/v1/W18-6507>. <https://aclanthology.org/W18-6507>
 33. Castro Ferreira, T., van der Lee, C., van Miltenburg, E., Krahmer, E.: Neural data-to-text generation: a comparison between pipeline and end-to-end architectures. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 552–562. Association for Computational Linguistics, Hong Kong (2019). <https://doi.org/10.18653/v1/D19-1052>. <https://aclanthology.org/D19-1052>
 34. Chen, G., van Deemter, K., Lin, C.: SimpleNLG-ZH: a linguistic realisation engine for Mandarin. In: Proceedings of the 11th International Conference on Natural Language Generation, pp. 57–66. Association for Computational Linguistics, Tilburg University (2018). <https://doi.org/10.18653/v1/W18-6506>. <https://aclanthology.org/W18-6506>
 35. Chen, G., Same, F., van Deemter, K.: Neural referential form selection: generalisability and interpretability. *Comput. Speech Lang.* **79**, 101466 (2023). <https://doi.org/10.1016/j.csl.2022.101466>. <https://www.sciencedirect.com/science/article/pii/S0885230822000894>
 36. Chen, G., Yao, J.G.: A closer look at recent results of verb selection for data-to-text NLG. In: Proceedings of the 12th International Conference on Natural Language Generation, pp. 158–163. Association for Computational Linguistics, Tokyo (2019). <https://doi.org/10.18653/v1/W19-8622>. <https://aclanthology.org/W19-8622>
 37. Chen, P.Y., Das, P.: AI maintenance: a robustness perspective. *Computer* **56**(2), 48–56 (2023). <https://doi.org/10.1109/MC.2022.3218005>
 38. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: scaling language modeling with pathways (2022). <https://arxiv.org/abs/2204.02311>
 39. Ciora, C., Iren, N., Alikhani, M.: Examining covert gender bias: a case study in Turkish and English machine translation models. In: Proceedings of the 14th International Conference on Natural Language Generation, pp. 55–63. Association for Computational Linguistics, Aberdeen (2021). <https://aclanthology.org/2021.inlg-1.7>

40. Dale, R.: Navigating the text generation revolution: traditional data-to-text NLG companies and the rise of ChatGPT. *Nat. Lang. Eng.* **29**(4), 1188–1197 (2023). <https://doi.org/10.1017/S1351324923000347>
41. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cogn. Sci.* **19**(2), 233–263 (1995). [https://doi.org/10.1016/0364-0213\(95\)90018-7](https://doi.org/10.1016/0364-0213(95)90018-7). <https://www.sciencedirect.com/science/article/pii/S0364021395900187>
42. Davey, A.: The formalisation of discourse production. Ph.D. thesis, University of Edinburgh (1974)
43. Davis, B.: 97 Things Every Project Manager Should Know: Collective Wisdom from the Experts. O'Reilly Media, Inc., Sebastopol, CA, USA (2009)
44. De Leeuw, E.D., Hox, J., Dillman, D.: *International Handbook of Survey Methodology*. Routledge, Abingdon, UK (2012)
45. van Deemter, K.: *Computational Models of Referring: A Study in Cognitive Science*. MIT Press, Cambridge, Mass, USA (2016)
46. van Deemter, K., Reiter, E.: 420Lying and Computational Linguistics. In: *The Oxford Handbook of Lying*. Oxford University Press (2018). <https://doi.org/10.1093/oxfordhb/9780198736578.013.32>
47. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). <https://doi.org/10.18653/v1/N19-1423>. <https://aclanthology.org/N19-1423>
48. Diakopoulos, N.: *Automating the News: How Algorithms Are Rewriting the Media*. Harvard University Press (2019)
49. Diakopoulos, N., Cools, H., Li, C., Helberger, N., Kung, E., Rinehart, A., Gibbs, L.: Generative AI in journalism: the evolution of newswork and ethics in a generative information ecosystem (2024). <https://doi.org/10.13140/RG.2.2.31540.05765>
50. Dinan, E., Abercrombie, G., Bergman, A., Spruit, S., Hovy, D., Boureau, Y.L., Rieser, V.: SafetyKit: first aid for measuring safety in open-domain conversational systems. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4113–4133. Association for Computational Linguistics, Dublin (2022). <https://doi.org/10.18653/v1/2022.acl-long.284>. <https://aclanthology.org/2022.acl-long.284>
51. Dror, R., Baumer, G., Shlomov, S., Reichart, R.: The hitchhiker's guide to testing statistical significance in natural language processing. In: Gurevych, I., Miyao, Y. (eds.) *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1383–1392. Association for Computational Linguistics, Melbourne (2018). <https://doi.org/10.18653/v1/P18-1128>. <https://aclanthology.org/P18-1128>
52. Dušek, O., Novikova, J., Rieser, V.: Evaluating the state-of-the-art of end-to-end natural language generation: the e2e NLG challenge. *Comput. Speech Lang.* **59**, 123–156 (2020). <https://www.sciencedirect.com/science/article/pii/S0885230819300919>
53. Dušek, O., Howcroft, D.M., Rieser, V.: Semantic noise matters for neural natural language generation. In: *Proceedings of the 12th International Conference on Natural Language Generation*, pp. 421–426. Association for Computational Linguistics, Tokyo (2019). <https://doi.org/10.18653/v1/W19-8652>. <https://aclanthology.org/W19-8652>
54. Elhadad, M., Robin, J.: An overview of SURGE: a reusable comprehensive syntactic realization component. In: *Eighth International Natural Language Generation Workshop (Posters and Demonstrations)* (1996). <https://aclanthology.org/W96-0501>
55. Farahani, A., Voghoei, S., Rasheed, K., Arabnia, H.R.: A brief review of domain adaptation. In: Stahlbock, R., Weiss, G.M., Abou-Nasr, M., Yang, C.Y., Arabnia, H.R., Deligiannidis, L. (eds.) *Advances in Data Science and Information Engineering*, pp. 877–894. Springer International Publishing, Cham (2021)

56. Feiner, S.K., McKeown, K.R.: Automating the generation of coordinated multimedia explanations. *Computer* **24**(10), 33–41 (1991)
57. Feng, S.Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., Hovy, E.: A survey of data augmentation approaches for NLP. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 968–988. Association for Computational Linguistics, Online (2021). <https://doi.org/10.18653/v1/2021.findings-acl.84>. <https://aclanthology.org/2021.findings-acl.84>
58. Field, A., Hole, G.: How to Design and Report Experiments. Sage, Thousand Oaks, California, United States (2003)
59. Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., Macherey, W.: Experts, errors, and context: a large-scale study of human evaluation for machine translation. *Trans. Assoc. Comput. Linguist.* **9**, 1460–1474 (2021). https://doi.org/10.1162/tacl_a_00437
60. Freitag, M., Grangier, D., Caswell, I.: BLEU might be guilty but references are not innocent. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 61–71. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.emnlp-main.5>. <https://aclanthology.org/2020.emnlp-main.5>
61. Freitag, M., Rei, R., Mathur, N., Lo, C.K., Stewart, C., Avramidis, E., Kocmi, T., Foster, G., Lavie, A., Martins, A.F.T.: Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In: Proceedings of the Seventh Conference on Machine Translation (WMT), pp. 46–68. Association for Computational Linguistics, Abu Dhabi (2022). <https://aclanthology.org/2022.wmt-1.2>
62. Ganguli, D., Lovitt, L., Kernion, J., Askell, A., Bai, Y., Kadavath, S., Mann, B., Perez, E., Schiefer, N., Ndousse, K., Jones, A., Bowman, S., Chen, A., Conerly, T., DasSarma, N., Drain, D., Elhage, N., El-Showk, S., Fort, S., Hatfield-Dodds, Z., Henighan, T., Hernandez, D., Hume, T., Jacobson, J., Johnston, S., Kravec, S., Olsson, C., Ringer, S., Tran-Johnson, E., Amodei, D., Brown, T., Joseph, N., McCandlish, S., Olah, C., Kaplan, J., Clark, J.: Red teaming language models to reduce harms: methods, scaling behaviors, and lessons learned (2022). <https://arxiv.org/abs/2209.07858>
63. Garg, T., Masud, S., Suresh, T., Chakraborty, T.: Handling bias in toxic speech detection: a survey. *ACM Comput. Surv.* **55**(13s) (2023). <https://doi.org/10.1145/3580494>
64. Garneau, N., Lamontagne, L.: Shared task in evaluating accuracy: leveraging pre-annotations in the validation process. In: Belz, A., Fan, A., Reiter, E., Sripada, Y. (eds.) Proceedings of the 14th International Conference on Natural Language Generation, pp. 266–270. Association for Computational Linguistics, Aberdeen (2021). <https://doi.org/10.18653/v1/2021.inlg-1.26>. <https://aclanthology.org/2021.inlg-1.26>
65. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *J. Artif. Intell. Res.* **61**, 65–170 (2018)
66. Gatt, A., Reiter, E.: SimpleNLG: a realisation engine for practical applications. In: Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), pp. 90–93. Association for Computational Linguistics, Athens (2009). <https://aclanthology.org/W09-0613>
67. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J.W., Wallach, H., Iii, H.D., Crawford, K.: Datasheets for datasets. *Commun. ACM* **64**(12), 86–92 (2021)
68. Gehrmann, S., Clark, E., Sellam, T.: Repairing the cracked foundation: a survey of obstacles in evaluation practices for generated text. *J. Artif. Intell. Res.* **77**, 103–166 (2023)
69. Gilbert, S., Harvey, H., Melvin, T., Vollebregt, E., Wicks, P.: Large language model AI chatbots require approval as medical devices. *Nat Med* **29**, 2396–2398 (2023). <https://doi.org/10.1038/s41591-023-02412-6>
70. Gkatzia, D., Lemon, O., Rieser, V.: Natural language generation enhances human decision-making with uncertain information. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 264–268. Association for Computational Linguistics, Berlin (2016). <https://doi.org/10.18653/v1/P16-2043>. <https://aclanthology.org/P16-2043>

71. Goldberg, E., Driedger, N., Kittredge, R.: Using natural-language processing to produce weather forecasts. *IEEE Expert* **9**(2), 45–53 (1994). <https://doi.org/10.1109/64.294135>
72. Goldberg, Y.: *Neural Network Methods for Natural Language Processing*. Springer Nature, Berlin (2022)
73. Goldman, N.: *Computer generation of natural-language from a deep conceptual base*. Ph.D. thesis, Stanford (1974)
74. Gonzalez-Corbelle, J., Alonso-Moral, J., Bugarín-Diz, A.: Some lessons learned reproducing human evaluation of a data-to-text system. In: *Proceedings of the 2023 Workshop on Human Evaluation* (2023). <https://aclanthology.org/2023.humeval-1.5/>
75. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press (2016). <http://www.deeplearningbook.org>
76. Gupta, S., Kohavi, R., Tang, D., Xu, Y., Andersen, R., Bakshy, E., Cardin, N., Chandran, S., Chen, N., Coey, D., et al.: Top challenges from the first practical online controlled experiments summit. *ACM SIGKDD Explorations Newsl.* **21**(1), 20–35 (2019)
77. Harbusch, K., Kempen, G.: Generating clausal coordinate ellipsis multilingually: a uniform approach based on postediting. In: *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pp. 138–145. Association for Computational Linguistics, Athens (2009). <https://aclanthology.org/W09-0624>
78. Heidari, P., Einolghozati, A., Jain, S., Batra, S., Callender, L., Arun, A., Mei, S., Gupta, S., Donmez, P., Bhardwaj, V., Kumar, A., White, M.: Getting to production with few-shot natural language generation models. In: *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 66–76. Association for Computational Linguistics, Singapore and Online (2021). <https://aclanthology.org/2021.sigdial-1.8>
79. Herbrich, R., Minka, T., Graepel, T.: Trueskill™: a Bayesian skill rating system. *Adv. Neural Inf. Process. Syst.* **19**, 569–576 (2006)
80. Hommes, S., van der Lee, C., Clouth, F., Vermunt, J., Verbeek, X., Krahmer, E.: A personalized data-to-text support tool for cancer patients. In: van Deemter, K., Lin, C., Takamura, H. (eds.) *Proceedings of the 12th International Conference on Natural Language Generation*, pp. 443–452. Association for Computational Linguistics, Tokyo (2019). <https://doi.org/10.18653/v1/W19-8656>. <https://aclanthology.org/W19-8656>
81. Howcroft, D.M., Belz, A., Clinciu, M.A., Gkatzia, D., Hasan, S.A., Mahamood, S., Mille, S., van Miltenburg, E., Santhanam, S., Rieser, V.: Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In: Davis, B., Graham, Y., Kelleher, J., Sripada, Y. (eds.) *Proceedings of the 13th International Conference on Natural Language Generation*, pp. 169–182. Association for Computational Linguistics, Dublin (2020). <https://doi.org/10.18653/v1/2020.inlg-1.23>. <https://aclanthology.org/2020.inlg-1.23>
82. Hu, Y., Jacob, J., Parker, G.J., Hawkes, D.J., Hurst, J.R., Stoyanov, D.: The challenges of deploying artificial intelligence models in a rapidly evolving pandemic. *Nat. Mach. Intell.* **2**(6), 298–300 (2020)
83. Hunter, J., Freer, Y., Gatt, A., Reiter, E., Sripada, S., Sykes, C.: Automatic generation of natural language nursing shift summaries in neonatal intensive care: Bt-nurse. *Artif. Intell. Med.* **56**(3), 157–172 (2012). <https://doi.org/10.1016/j.artmed.2012.09.002>. <https://www.sciencedirect.com/science/article/pii/S0933365712001170>
84. Hüske-Kraus, D.: Text generation in clinical medicine—a review. *Methods Inf. Med.* **42**(01), 51–60 (2003)
85. Ioannidis, J.P.: Why most published research findings are false. *PLoS Med.* **2**(8), e124 (2005)
86. Johnson, A.E., Pollard, T.J., Shen, L., Lehman, L.W.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., Mark, R.G.: Mimic-III, a freely accessible critical care database. *Sci. Data* **3**(1), 1–9 (2016)
87. Jones, K.S., Galliers, J.R.: *Evaluating Natural Language Processing Systems: An Analysis and Review* (Lecture Notes in Computer Science, 1083). Springer (1996)

88. Jurafsky, D., Martin, J.H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd edn. Prentice Hall, London (2008)
89. Kale, M., Rastogi, A.: Template guided text generation for task-oriented dialogue. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6505–6520. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.emnlp-main.527>. <https://aclanthology.org/2020.emnlp-main.527>
90. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 289–296 (2001). <https://doi.org/10.1109/ICDM.2001.989531>
91. Kibble, R., Power, R.: Optimizing referential coherence in text generation. *Comput. Linguist.* **30**(4), 401–416 (2004). <https://doi.org/10.1162/0891201042544893>. <https://aclanthology.org/J04-4001>
92. Kincaid, J.P., Fishburne Jr, R.P., Rogers, R.L., Chissom, B.S.: Derivation of new readability formulas (automated readability index, fog count and Flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch (1975)
93. Kittredge, R., Iordanskaja, L., Polguère, A.: Multi-lingual text generation and the meaning-text theory. In: *Proceedings of the Second Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Pittsburgh (1988). <https://aclanthology.org/1988.tmi-1.5>
94. Kittredge, R., Lehrberger, J.: *Sublanguage*. de Gruyter, Berlin (1982)
95. Knoll, T., Moramarco, F., Papadopoulos Korfiatis, A., Young, R., Ruffini, C., Perera, M., Perstl, C., Reiter, E., Belz, A., Savkov, A.: User-driven research of medical note generation software. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 385–394. Association for Computational Linguistics, Seattle (2022). <https://doi.org/10.18653/v1/2022.naacl-main.29>. <https://aclanthology.org/2022.naacl-main.29>
96. Kocmi, T., Bawden, R., Bojar, O., Dvorkovich, A., Federmann, C., Fishel, M., Gowda, T., Graham, Y., Grundkiewicz, R., Haddow, B., Knowles, R., Koehn, P., Monz, C., Morishita, M., Nagata, M., Nakazawa, T., Novák, M., Popel, M., Popović, M.: Findings of the 2022 conference on machine translation (WMT22). In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pp. 1–45. Association for Computational Linguistics, Abu Dhabi (2022). <https://aclanthology.org/2022.wmt-1.1>
97. Kocmi, T., Federmann, C.: GEMBA-MQM: detecting translation quality error spans with GPT-4. In: Koehn, P., Haddow, B., Kocmi, T., Monz, C. (eds.) *Proceedings of the Eighth Conference on Machine Translation*, pp. 768–775. Association for Computational Linguistics, Singapore (2023). <https://doi.org/10.18653/v1/2023.wmt-1.64>. <https://aclanthology.org/2023.wmt-1.64>
98. Kocmi, T., Federmann, C., Grundkiewicz, R., Junczys-Dowmunt, M., Matsushita, H., Menezes, A.: To ship or not to ship: an extensive evaluation of automatic metrics for machine translation. In: *Proceedings of the Sixth Conference on Machine Translation*, pp. 478–494. Association for Computational Linguistics, Online (2021). <https://aclanthology.org/2021.wmt-1.57>
99. Koehn, P.: Statistical significance tests for machine translation evaluation. In: Lin, D., Wu, D. (eds.) *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395. Association for Computational Linguistics, Barcelona (2004). <https://aclanthology.org/W04-3250>
100. Koehn, P.: *Neural Machine Translation*. Cambridge University Press, Cambridge, UK (2020)
101. Kondadadi, R., Howald, B., Schilder, F.: A statistical NLG framework for aggregated planning and realization. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1406–1415. Association for Computational Linguistics, Sofia (2013). <https://aclanthology.org/P13-1138>

102. Krahmer, E., van Deemter, K.: Computational generation of referring expressions: a survey. *Comput. Linguist.* **38**(1), 173–218 (2012). https://doi.org/10.1162/COLI_a_00088
103. Kukich, K.: Design of a knowledge-based report generator. In: 21st Annual Meeting of the Association for Computational Linguistics, pp. 145–150. Association for Computational Linguistics, Cambridge, MA (1983). <https://doi.org/10.3115/981311.981340>. <https://aclanthology.org/P83-1022>
104. Kukich, K.: Where do phrases come from: Some preliminary experiments in connectionist phrase generation. In: *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pp. 405–421. Springer, Berlin (1987)
105. Lakens, D.: Improving your statistical inferences. <https://doi.org/10.5281/zenodo.6409077>. https://lakens.github.io/statistical_inferences/
106. Langkilde, I., Knight, K.: Generation that exploits corpus-based statistical knowledge. In: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, vol. 1, pp. 704–710. Association for Computational Linguistics, Montreal (1998). <https://doi.org/10.3115/980845.980963>. <https://aclanthology.org/P98-1116>
107. Lapalme, G.: Data-to-text bilingual generation (2023). <https://arxiv.org/abs/2311.14808>
108. Lauer, T., et al.: *Change Management*. Springer, Berlin (2010)
109. Law, A.S., Freer, Y., Hunter, J., Logie, R.H., McIntosh, N., Quinn, J.: A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit. *J. Clin. Monitor. Comput.* **19**, 183–194 (2005)
110. Lennox, A.S., Osman, L.M., Reiter, E., Robertson, R., Friend, J., McCann, I., Skatun, D., Donnan, P.T.: Cost effectiveness of computer tailored and non-tailored smoking cessation letters in general practice: randomised controlled trial. *BMJ* **322**(7299), 1396 (2001). <https://doi.org/10.1136/bmj.322.7299.1396>. <https://www.bmj.com/content/322/7299/1396>
111. Leppänen, L., Munezero, M., Granroth-Wilding, M., Toivonen, H.: Data-driven news generation for automated journalism. In: *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 188–197. Association for Computational Linguistics, Santiago de Compostela (2017). <https://doi.org/10.18653/v1/W17-3528>. <https://aclanthology.org/W17-3528>
112. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Doklady* **10**, 707 (1966)
113. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.acl-main.703>. <https://aclanthology.org/2020.acl-main.703>
114. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*. Curran Associates Inc., Red Hook (2020)
115. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: *Text Summarization Branches Out*, pp. 74–81. Association for Computational Linguistics, Barcelona (2004). <https://aclanthology.org/W04-1013>
116. Lu, Y., Shen, M., Wang, H., Wang, X., van Rechem, C., Wei, W.: Machine learning for synthetic data generation: a review (2024). <https://arxiv.org/abs/2302.04062>
117. Luhn, H.P.: A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.* **1**(4), 309–317 (1957). <https://doi.org/10.1147/rd.14.0309>
118. Magesh, V., Surani, F., Dahl, M., Suzgun, M., Manning, C.D., Ho, D.E.: Hallucination-free? Assessing the reliability of leading AI legal research tools (2024). <https://arxiv.org/abs/2405.20362>
119. Mahamood, S., Bradshaw, W., Reiter, E.: Generating annotated graphs using the NLG pipeline architecture. In: Mitchell, M., McCoy, K., McDonald, D., Cahill, A. (eds.) *Proceedings of the*

- 8th International Natural Language Generation Conference (INLG), pp. 123–127. Association for Computational Linguistics, Philadelphia (2014). <https://doi.org/10.3115/v1/W14-4419>. <https://aclanthology.org/W14-4419>
120. Mahamood, S., Reiter, E.: Generating affective natural language for parents of neonatal infants. In: Proceedings of the 13th European Workshop on Natural Language Generation, pp. 12–21. Association for Computational Linguistics, Nancy (2011). <https://aclanthology.org/W11-2803>
 121. Mahamood, S., Reiter, E.: Working with clinicians to improve a patient-information NLG system. In: INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference, pp. 100–104. Association for Computational Linguistics, Utica (2012). <https://aclanthology.org/W12-1516>
 122. Mani, I., House, D., Klein, G., Hirschman, L., Firmin, T., Sundheim, B.: The TIPSTER SUMMAC text summarization evaluation. In: Ninth Conference of the European Chapter of the Association for Computational Linguistics, pp. 77–85. Association for Computational Linguistics, Bergen (1999). <https://aclanthology.org/E99-1011>
 123. Mani, I., Klein, G., House, D., Hirschman, L., Firmin, T., Sundheim, B.: Summac: a text summarization evaluation. *Nat. Lang. Eng.* **8**(1), 43–68 (2002)
 124. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: a theory of text organization. University of Southern California, Information Sciences Institute, Los Angeles (1987)
 125. Marsh, E., Hamburger, H., Grishman, R.: A production rule system for message summarization. In: Proceedings of the Fourth AAAI Conference on Artificial Intelligence, AAAI'84, pp. 243–246. AAAI Press (1984)
 126. Mathur, N., Baldwin, T., Cohn, T.: Tangled up in BLEU: reevaluating the evaluation of automatic machine translation evaluation metrics. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4984–4997. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.acl-main.448>. <https://aclanthology.org/2020.acl-main.448>
 127. McHugh, M.L.: Interrater reliability: the kappa statistic. *Biochemia Medica* **22**(3), 276–282 (2012)
 128. McKeown, K.R., Feiner, S.K., Dalal, M., Chang, S.F.: Generating multimedia briefings: coordinating language and illustration. *Artif. Intell.* **103**(1), 95–116 (1998). [https://doi.org/10.1016/S0004-3702\(98\)00072-1](https://doi.org/10.1016/S0004-3702(98)00072-1). <https://www.sciencedirect.com/science/article/pii/S0004370298000721>. Artificial Intelligence 40 years later
 129. McKinlay, A., McVittie, C., Reiter, E., Freer, Y., Sykes, C., Logie, R.: Design issues for socially intelligent user interfaces. *Methods Inf. Med.* **49**(04), 379–387 (2010)
 130. Meehl, P.E.: Clinical versus statistical prediction: a theoretical analysis and a review of the evidence (1954)
 131. Melin, M., Bäck, A., SöDergård, C., Munezero, M.D., Leppänen, L.J., Toivonen, H.: No landslide for the human journalist – an empirical study of computer-generated election news in Finland. *IEEE Access* **6**, 43356–43367 (2018). <https://doi.org/10.1109/ACCESS.2018.2861987>
 132. Mille, S., Belz, A., Bohnet, B., Castro Ferreira, T., Graham, Y., Wanner, L.: The third multilingual surface realisation shared task (SR'20): overview and evaluation results. In: Proceedings of the Third Workshop on Multilingual Surface Realisation, pp. 1–20. Association for Computational Linguistics, Barcelona (Online) (2020). <https://aclanthology.org/2020.msr-1.1>
 133. van Miltenburg, E., Clinciu, M., Dušek, O., Gkatzia, D., Inglis, S., Leppänen, L., Mahamood, S., Manning, E., Schoch, S., Thomson, C., Wen, L.: Underreporting of errors in NLG output, and what to do about it. In: Belz, A., Fan, A., Reiter, E., Sripada, Y. (eds.) Proceedings of the 14th International Conference on Natural Language Generation, pp. 140–153. Association for Computational Linguistics, Aberdeen (2021). <https://doi.org/10.18653/v1/2021.inlg-1.14>. <https://aclanthology.org/2021.inlg-1.14>
 134. van Miltenburg, E., van der Lee, C., Krahmer, E.: Preregistering NLP research. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R.,

- Chakraborty, T., Zhou, Y. (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 613–623. Association for Computational Linguistics, Online (2021). <https://doi.org/10.18653/v1/2021.naacl-main.51>. <https://aclanthology.org/2021.naacl-main.51>
135. Moher, D., Liberati, A., Tetzlaff, J., Altman, D.: Group prisma (2009) preferred reporting items for systematic reviews and metaanalyses: the PRISMA statement. *BMJ* **339**(21), b2535 (2009)
 136. Moncur, W., Masthoff, J., Reiter, E., Freer, Y., Nguyen, H.: Providing adaptive health updates across the personal social network. *Hum.–Comput. Interact.* **29**(3), 256–309 (2014). <https://doi.org/10.1080/07370024.2013.819218>
 137. Moramarco, F.: Evaluation of medical note generation systems. Ph.D. thesis, University of Aberdeen (2024)
 138. Moramarco, F., Papadopoulos Korfiatis, A., Perera, M., Juric, D., Flann, J., Reiter, E., Belz, A., Savkov, A.: Human evaluation and correlation with automatic metrics in consultation note generation. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5739–5754. Association for Computational Linguistics, Dublin (2022). <https://doi.org/10.18653/v1/2022.acl-long.394>. <https://aclanthology.org/2022.acl-long.394>
 139. Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., Fernández-Leal, Á.: Human-in-the-loop machine learning: a state of the art. *Artif. Intell. Rev.* **56**(4), 3005–3054 (2023)
 140. Nielsen, J.: Usability Engineering. Morgan Kaufmann, Burlington, Massachusetts, United States (1994)
 141. Nunberg, G.: The linguistics of punctuation. 18. Center for the Study of Language (CSLI) (1990)
 142. O’Brien, S.: How to deal with errors in machine translation: Postediting. *Mach. Transl. Everyone: Empowering Users Age Artif. Intell.* **18**, 105 (2022)
 143. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. *Adv. Neural Inf. Process. Syst.* **35**, 27730–27744 (2022)
 144. Papadopoulos Korfiatis, A., Moramarco, F., Sarac, R., Savkov, A.: Primock57: a dataset of primary care mock consultations. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (2022)
 145. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics, Philadelphia (2002). <https://doi.org/10.3115/1073083.1073135>. <https://aclanthology.org/P02-1040>
 146. Peter, J.: Artificial Versifying; or, the Schoolboy’s Recreation. A new way to make Latin verses etc. For John Sims, London (1677)
 147. Popović, M.: chrF: character n-gram F-score for automatic MT evaluation. In: Proceedings of the Tenth Workshop on Statistical Machine Translation, pp. 392–395. Association for Computational Linguistics, Lisbon (2015). <https://doi.org/10.18653/v1/W15-3049>. <https://aclanthology.org/W15-3049>
 148. Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., Sykes, C.: Automatic generation of textual summaries from neonatal intensive care data. *Artif. Intell.* **173**(7), 789–816 (2009). <https://doi.org/10.1016/j.artint.2008.12.002>. <https://www.sciencedirect.com/science/article/pii/S0004370208002117>
 149. Post, M.: A call for clarity in reporting BLEU scores. In: Bojar, O., Chatterjee, R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Yepes, A.J., Koehn, P., Monz, C., Negri, M., N  v  ol, A., Neves, M., Post, M., Specia, L., Turchi, M., Verspoor, K. (eds.) Proceedings of the Third Conference on Machine Translation: Research Papers, pp. 186–191. Association for Computational Linguistics, Brussels (2018). <https://doi.org/10.18653/v1/W18-6319>. <https://aclanthology.org/W18-6319>

150. Prasad, V., Kim, C., Burotto, M., Vandross, A.: The strength of association between surrogate end points and survival in oncology: a systematic review of trial-level meta-analyses. *JAMA Internal Med.* **175**(8), 1389–1398 (2015)
151. Puduppully, R., Fu, Y., Lapata, M.: Data-to-text generation with variational sequential planning. *Trans. Assoc. Comput. Linguist.* **10**, 697–715 (2022). <https://aclanthology.org/2022.tacl-1.40>
152. Puduppully, R., Lapata, M.: Data-to-text generation with macro planning. *Trans. Assoc. Comput. Linguist.* **9**, 510–527 (2021). <https://aclanthology.org/2021.tacl-1.31>
153. Radev, D.R., Hovy, E., McKeown, K.: Introduction to the special issue on summarization. *Computat. Linguist.* **28**(4), 399–408 (2002). <https://doi.org/10.1162/089120102762671927>. <https://aclanthology.org/J02-4001>
154. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**(140), 1–67 (2020)
155. Ramos-Soto, A., Reiter, E., van Deemter, K., Alonso, J., Gatt, A.: Meteorologists and students: a resource for language grounding of geographical descriptors. In: Proceedings of the 11th International Conference on Natural Language Generation, pp. 421–425. Association for Computational Linguistics, Tilburg University (2018). <https://doi.org/10.18653/v1/W18-6551>. <https://aclanthology.org/W18-6551>
156. Ramponi, A., Plank, B.: Neural unsupervised domain adaptation in NLP—a survey. In: Scott, D., Bel, N., Zong, C. (eds.) Proceedings of the 28th International Conference on Computational Linguistics, pp. 6838–6855. International Committee on Computational Linguistics, Barcelona (Online) (2020). <https://doi.org/10.18653/v1/2020.coling-main.603>. <https://aclanthology.org/2020.coling-main.603>
157. Reiter, E.: Generating appropriate natural language object descriptions. Ph.D. thesis, Harvard (1990)
158. Reiter, E.: An architecture for data-to-text systems. In: Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07), pp. 97–104. DFKI GmbH, Saarbrücken (2007). <https://aclanthology.org/W07-2315>
159. Reiter, E.: A commercial perspective on reference. In: Proceedings of the 10th International Conference on Natural Language Generation, pp. 134–138. Association for Computational Linguistics, Santiago de Compostela (2017). <https://doi.org/10.18653/v1/W17-3519>. <https://aclanthology.org/W17-3519>
160. Reiter, E.: A structured review of the validity of BLEU. *Comput. Linguist.* **44**(3), 393–401 (2018). https://doi.org/10.1162/coli_a_00322
161. Reiter, E., Belz, A.: An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computat. Linguist.* **35**(4), 529–558 (2009). <https://doi.org/10.1162/coli.2009.35.4.35405>. <https://aclanthology.org/J09-4008>
162. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, Cambridge, UK (2000)
163. Reiter, E., Gatt, A., Portet, F., van der Meulen, M.: The importance of narrative and other lessons from an evaluation of an NLG system that summarises clinical data. In: Proceedings of the Fifth International Natural Language Generation Conference, pp. 147–156. Association for Computational Linguistics, Salt Fork (2008). <https://aclanthology.org/W08-1119>
164. Reiter, E., Mellish, C., Levine, J.: Automatic generation of technical documentation. *Appl. Artif. Intell. Int. J.* **9**(3), 259–287 (1995)
165. Reiter, E., Robertson, R., Osman, L.M.: Lessons from a failure: generating tailored smoking cessation letters. *Artif. Intell.* **144**(1), 41–58 (2003). [https://doi.org/10.1016/S0004-3702\(02\)00370-3](https://doi.org/10.1016/S0004-3702(02)00370-3). <https://www.sciencedirect.com/science/article/pii/S0004370202003703>
166. Reiter, E., Sripada, S.: Human variation and lexical choice. *Comput. Linguist.* **28**(4), 545–553 (2002). <https://doi.org/10.1162/089120102762671981>
167. Reiter, E., Sripada, S., Hunter, J., Yu, J., Davy, I.: Choosing words in computer-generated weather forecasts. *Artif. Intell.* **167**(1), 137–169 (2005). <https://www.sciencedirect.com/science/article/pii/S0004370205000998>. Connecting Language to the World

168. Reiter, E., Sripada, S.G., Robertson, R.: Acquiring correct knowledge for natural language generation. *J. Artif. Int. Res.* **18**(1), 491–516 (2003)
169. Rennard, V., Shang, G., Hunter, J., Vazirgiannis, M.: Abstractive meeting summarization: a survey. *Trans. Assoc. Comput. Linguist.* **11**, 861–884 (2023). https://doi.org/10.1162/tacl_a_00578
170. Ribeiro, M.T., Wu, T., Guestrin, C., Singh, S.: Beyond accuracy: behavioral testing of NLP models with CheckList. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.acl-main.442>. <https://aclanthology.org/2020.acl-main.442>
171. Rogers, A.: Changing the world by changing the data. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2182–2194. Association for Computational Linguistics, Online (2021). <https://doi.org/10.18653/v1/2021.acl-long.170>. <https://aclanthology.org/2021.acl-long.170>
172. Rogers, Y., Sharp, H., Preece, J.: *Interaction Design: Beyond Human-Computer Interaction*, 6th edn. Wiley, Hoboken, New Jersey, U.S (2023)
173. Sainz, O., Campos, J., Garcia-Ferrero, I., Etxaniz, J., de Lacalle, O.L., Agirre, E.: NLP evaluation in trouble: on the need to measure LLM data contamination for each benchmark. In: Bouamor, H., Pino, J., Bali, K. (eds.) *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787. Association for Computational Linguistics, Singapore (2023). <https://doi.org/10.18653/v1/2023.findings-emnlp.722>. <https://aclanthology.org/2023.findings-emnlp.722>
174. Sakaguchi, K., Post, M., Van Durme, B.: Efficient elicitation of annotations for human evaluation of machine translation. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 1–11. Association for Computational Linguistics, Baltimore (2014). <https://doi.org/10.3115/v1/W14-3301>. <https://aclanthology.org/W14-3301>
175. Sakai, T., Nagao, M.: Sentence generation by semantic concordance. In: *COLING 1965* (1965). <https://aclanthology.org/C65-1022>
176. Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A.S., Yvon, F., Gallé, M., et al.: Bloom: a 176b-parameter open-access multilingual language model (2022). <https://arxiv.org/abs/2211.05100>
177. Schaeffer, R.: Pretraining on the test set is all you need (2023). <https://arxiv.org/abs/2309.08632>
178. Department for Science, I., Technology: *International scientific report on the safety of advanced AI* (2024). <https://www.gov.uk/government/publications/international-scientific-report-on-the-safety-of-advanced-ai>
179. Sellam, T., Das, D., Parikh, A.: BLEURT: learning robust metrics for text generation. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7881–7892. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.acl-main.704>. <https://aclanthology.org/2020.acl-main.704>
180. Shimorina, A., Belz, A.: The human evaluation datasheet: a template for recording details of human evaluation experiments in NLP. In: *Proceedings of the 2nd Workshop on Human Evaluation of NLP Systems (HumEval)*, pp. 54–75. Association for Computational Linguistics, Dublin (2022). <https://doi.org/10.18653/v1/2022.humeval-1.6>. <https://aclanthology.org/2022.humeval-1.6>
181. Silberman, M.S., Tomlinson, B., LaPlante, R., Ross, J., Irani, L., Zaldivar, A.: Responsible research with crowds: pay crowdworkers at least minimum wage. *Commun. ACM* **61**(3), 39–41 (2018). <https://doi.org/10.1145/3180492>
182. Singhal, K., Azizi, S., Tu, T., Mahdavi, S.S., Wei, J., Chung, H.W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al.: Large language models encode clinical knowledge. *Nature* **620**(7972), 172–180 (2023)

183. Sivaprasad, A., Reiter, E.: Linguistically communicating uncertainty in patient-facing risk prediction models. In: Vázquez, R., Celikkanat, H., Ulmer, D., Tiedemann, J., Swayamdipta, S., Aziz, W., Plank, B., Baan, J., de Marneffe, M.C. (eds.) *Proceedings of the 1st Workshop on Uncertainty-Aware NLP (UncertainNLP 2024)*, pp. 127–132. Association for Computational Linguistics, St Julians (2024). <https://aclanthology.org/2024.uncertainlp-1.13>
184. Smiley, C., Schilder, F., Plachouras, V., Leidner, J.L.: Say the right thing right: ethics issues in natural language generation systems. In: Hovy, D., Spruit, S., Mitchell, M., Bender, E.M., Strube, M., Wallach, H. (eds.) *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pp. 103–108. Association for Computational Linguistics, Valencia (2017). <https://doi.org/10.18653/v1/W17-1613>. <https://aclanthology.org/W17-1613>
185. Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J.W., Kreps, S., et al.: Release strategies and the social impacts of language models (2019). <https://arxiv.org/abs/1908.09203>
186. Sripada, S., Burnett, N., Turner, R., Mastin, J., Evans, D.: A case study: NLG meeting weather industry demand for quality and quantity of textual weather forecasts. In: *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pp. 1–5. Association for Computational Linguistics, Philadelphia (2014). <https://doi.org/10.3115/v1/W14-4401>. <https://aclanthology.org/W14-4401>
187. Sripada, S., Reiter, E., Hawizy, L.: Evaluation of an NLG system using post-edit data: lessons learnt. In: *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*. Association for Computational Linguistics, Aberdeen (2005). <https://aclanthology.org/W05-1615>
188. Sripada, S.G., Reiter, E., Hunter, J., Yu, J.: Generating English summaries of time series data using the Gricean maxims. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD’03*, pp. 187–196. Association for Computing Machinery, New York (2003). <https://doi.org/10.1145/956750.956774>
189. Stangl, A., Morris, M.R., Gurari, D.: “person, shoes, tree. is the person naked?” what people with vision impairments want in image descriptions. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI’20*, pp. 1–13. Association for Computing Machinery, New York (2020). <https://doi.org/10.1145/3313831.3376404>
190. Strickland, E.: IBM Watson, heal thyself: How IBM overpromised and underdelivered on AI health care. *IEEE Spect.* **56**(4), 24–31 (2019)
191. Sun, M., Reiter, E., Kiltie, A.E., Ramsay, G., Duncan, L., Murchie, P., Adam, R.: Effectiveness of ChatGPT in explaining complex medical reports to patients (2024). <https://arxiv.org/abs/2406.15963>
192. Syed, M.: *Black Box Thinking: Why Most People Never Learn from Their Mistakes—But Some Do*. Penguin (2015)
193. Tamayo-Sarver, J.: I’m an ER doctor: Here’s what I found when I asked chatgpt to diagnose my patients (2023). <https://inflecthealth.medium.com/im-an-er-doctor-here-s-what-i-found-when-i-asked-chatgpt-to-diagnose-my-patients-7829c375a9da>. Retrieved on 8 Sep 2023
194. Thomson, C., Reiter, E.: A gold standard methodology for evaluating accuracy in data-to-text systems. In: *Proceedings of the 13th International Conference on Natural Language Generation*, pp. 158–168. Association for Computational Linguistics, Dublin (2020). <https://aclanthology.org/2020.inlg-1.22>
195. Thomson, C., Reiter, E., Belz, A.: Common flaws in running human evaluation experiments in NLP. *Comput. Linguist.* **50**, 795–805 (2024). <https://aclanthology.org/2024.cl-2.9/>
196. Thomson, C., Reiter, E., Sripada, S.: Comprehension driven document planning in natural language generation systems. In: *Proceedings of the 11th International Conference on Natural Language Generation*, pp. 371–380. Association for Computational Linguistics, Tilburg University (2018). <https://doi.org/10.18653/v1/W18-6544>. <https://aclanthology.org/W18-6544>
197. Thomson, C., Reiter, E., Sundararajan, B.: Evaluating factual accuracy in complex data-to-text. *Comput. Speech Lang.* **80**, 101482 (2023). <https://doi.org/10.1016/j.csl.2023.101482>. <https://www.sciencedirect.com/science/article/pii/S0885230823000013>

198. Tintarev, N., Reiter, E., Black, R., Waller, A., Reddington, J.: Personal storytelling: using natural language generation for children with complex communication needs, in the wild. . . . *Int. J. Hum.-Comput. Stud.* **92-93**, 1–16 (2016). <https://doi.org/10.1016/j.ijhcs.2016.04.005>. <https://www.sciencedirect.com/science/article/pii/S1071581916300155>
199. Tukey, J.W., et al.: *Exploratory Data Analysis*, vol. 2. Reading, MA, Addison-Wesley (1977)
200. Tunstall, L., Von Werra, L., Wolf, T.: *Natural Language Processing with Transformers*, Revised Edition. O'Reilly Media, Incorporated (2022)
201. Turner, R., Sripada, Y., Reiter, E.: Generating approximate geographic descriptions. In: *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pp. 42–49. Association for Computational Linguistics, Athens (2009). <https://aclanthology.org/W09-0607>
202. Van Deemter, K.: *Not Exactly: In Praise of Vagueness*. OUP Oxford, Oxford (2010)
203. van der Lee, C., Gatt, A., van Miltenburg, E., Krahmer, E.: Human evaluation of automatically generated text: current trends and best practice guidelines. *Comput. Speech Lang.* **67**, 101151 (2021). <https://doi.org/10.1016/j.csl.2020.101151>. <https://www.sciencedirect.com/science/article/pii/S088523082030084X>
204. Van Der Meulen, M., Logie, R.H., Freer, Y., Sykes, C., McIntosh, N., Hunter, J.: When a graph is poorer than 100 words: a comparison of computerised natural language generation, human generated descriptions and graphical displays in neonatal intensive care. *Appl. Cogn. Psychol. Off. J. Soc. Appl. Res. Mem. Cogn.* **24**(1), 77–89 (2010)
205. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.U., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., Red Hook, New York, USA (2017)
206. Wahlster, W., André, E., Finkler, W., Profitlich, H.J., Rist, T.: Plan-based integration of natural language and graphics generation. *Artif. Intell.* **63**(1–2), 387–427 (1993)
207. Walker, M.A., Stent, A., Mairesse, F., Prasad, R.: Individual and domain adaptation in sentence planning for dialogue. *J. Artif. Intell. Res.* **30**, 413–456 (2007)
208. Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., Sui, Z.: Large language models are not fair evaluators. In: *Proceedings of The 62nd Annual Meeting of the Association for Computational Linguistics* (2024). <https://aclanthology.org/2024.acl-long.511/>
209. van de Water, L.F., Bos-van den Hoek, D.W., Kuijper, S.C., van Laarhoven, H.W., Creemers, G.J., Dohmen, S.E., Fiebrich, H.B., Ottevanger, P.B., Sommeijer, D.W., de Vos, F.Y., et al.: Potential adverse outcomes of shared decision making about palliative cancer treatment: a secondary analysis of a randomized trial. *Med. Decis. Making* **44**(1), 89–101 (2024)
210. Watson, L., Gkatzia, D.: Unveiling NLG human-evaluation reproducibility: Lessons learned and key insights from participating in the ReproNLP challenge. In: *Proceedings of the 2023 Workshop on Human Evaluation* (2023). <https://aclanthology.org/2023.humeval-1.6/>
211. Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners. In: *International Conference on Learning Representations*
212. Weidinger, L., Rauh, M., Marchal, N., Manzini, A., Hendricks, L.A., Mateos-Garcia, J., Bergman, S., Kay, J., Griffin, C., Bariach, B., Gabriel, I., Rieser, V., Isaac, W.: Sociotechnical safety evaluation of generative AI systems (2023). <https://arxiv.org/abs/2310.11986>
213. Weißgraeber, R., Madsack, A.: A working, non-trivial, topically indifferent NLG system for 17 languages. In: *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 156–157. Association for Computational Linguistics, Santiago de Compostela (2017). <https://doi.org/10.18653/v1/W17-3524>. <https://aclanthology.org/W17-3524>
214. Welbl, J., Glaese, A., Uesato, J., Dathathri, S., Mellor, J., Hendricks, L.A., Anderson, K., Kohli, P., Coppin, B., Huang, P.S.: Challenges in detoxifying language models. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2447–2469. Association for Computational Linguistics, Punta Cana (2021). <https://doi.org/10.18653/v1/2021.findings-emnlp.210>. <https://aclanthology.org/2021.findings-emnlp.210>

215. White, M., Howcroft, D.M.: Inducing clause-combining rules: a case study with the SPaRky restaurant corpus. In: Proceedings of the 15th European Workshop on Natural Language Generation (ENLG), pp. 28–37. Association for Computational Linguistics, Brighton (2015). <https://doi.org/10.18653/v1/W15-4704>. <https://aclanthology.org/W15-4704>
216. White, M., Rajkumar, R., Martin, S.: Towards broad coverage surface realization with CCG. In: Proceedings of the Workshop on Using Corpora for Natural Language Generation (2007)
217. Wieggers, K.E., Beatty, J.: Software Requirements. Pearson Education, London (2013)
218. Wieling, M., Rawee, J., van Noord, G.: Reproducibility in computational linguistics: are we willing to share? *Comput. Linguist.* **44**(4), 641–649 (2018). https://doi.org/10.1162/coli_a_00330
219. Wikipedia, T.F.E.: Replication crisis (2023). https://en.wikipedia.org/wiki/Replication_crisis. Retrieved on 11 July 2023
220. Williams, S., Reiter, E.: Generating basic skills reports for low-skilled readers. *Nat. Lang. Eng.* **14**(4), 495–525 (2008). <https://doi.org/10.1017/S1351324908004725>
221. Wiseman, S., Shieber, S., Rush, A.: Challenges in data-to-document generation. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2253–2263. Association for Computational Linguistics, Copenhagen (2017). <https://doi.org/10.18653/v1/D17-1239>. <https://aclanthology.org/D17-1239>
222. Wu, Z., Balloccu, S., Reiter, E., Helaoui, R., Reforgiato Recupero, D., Riboni, D.: Are experts needed? On human evaluation of counselling reflection generation. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6906–6930. Association for Computational Linguistics, Toronto (2023). <https://doi.org/10.18653/v1/2023.acl-long.382>. <https://aclanthology.org/2023.acl-long.382>
223. Zhang, L., Mille, S., Hou, Y., Deutsch, D., Clark, E., Liu, Y., Mahamood, S., Gehrmann, S., Clinciu, M., Chandu, K.R., Sedoc, J.: A needle in a haystack: an analysis of high-agreement workers on MTurk for summarization. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14944–14982. Association for Computational Linguistics, Toronto (2023). <https://doi.org/10.18653/v1/2023.acl-long.835>. <https://aclanthology.org/2023.acl-long.835>
224. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: evaluating text generation with bert. In: International Conference on Learning Representations (2020)

Index

A

A/B testing, 15, 134
Acceptability, 16, 144, 163, 168
Accessibility, 18
Accuracy, 11–13, 16, 21, 71, 74, 76, 93, 135, 139, 145, 161, 168, 174
Acronyms, 50
Adaptability, 161
Aggregation, 37, 40
Alignment, 59
Analysis, 99, 116, 120, 127
Analytics, 10, 32, 35
 articulate analytics, 32
Annotation, 108, 114, 118, 129
ANOVA, 117
Applications, 16
Appropriateness, 12
Arabic, 43
Arria, 4, 16, 20, 22, 47, 166, 169
Attention, 57
Attention check, 113, 116
Auditable, 10, 25, 68, 70
Ax Semantics, 20, 47

B

Babytalk, 29, 31, 33–35, 38, 40, 77, 96, 103, 112–114, 136–138, 146, 160, 177
 BT-Family, 29, 153, 164
 BT45, 29, 83, 109, 177
 BT-Nurse, 29, 146
Back-propagation, 57
Back-translate, 64
Bard, 147, 171

BART, 8, 11, 21, 52, 58
Baselines, 95, 101, 139
BasketballNLG, 110–112, 114–116, 118
BBC, 16
BBC election reporter, 160, 165
Behaviour change, 17, 175
Benefits, 16, 137
Berne convention, 69
BERT, 129
BERTScore, 122
Beta test, 88
Bias, 18, 63, 130, 132
Binomial test, 98, 117
BLEU, 20, 122, 127, 128, 131, 132, 139
BLEURT, 122, 129
BLOOM, 58, 65, 173
Brand, 155
 loyalty, 138
Bugs, 15, 138, 148, 151, 152, 155
 analysis, 100
 code, 100, 120
Business intelligence, 16, 85, 160, 169, 177

C

Change management, 91, 137, 163, 173
Chatbot, 85
ChatGPT, 1, 2, 11, 20, 53, 59, 63, 67, 146, 162, 178
Children, 160
Chi-square test, 98, 117
ChrF, 122, 129
Clarity, 73
Classifiers, 55

CoGenTex, 20, 22
 Cognitive science, 40, 97
 Coherence, 35
 Commercial evaluation, 136
 Common Crawl, 63
 Comparison experiment, 14
 Compute
 cost, 77, 124
 resources, 77, 124
 speed, 77, 124
 Conceptual aggregation, 41
 Configurability, 16, 40, 136
 Consistency, 78
 Consultations, 6, 135, 174
 Content, 35, 76
 Context, 39, 149
 Control groups, 95
 Controllability, 10, 11, 21, 25, 68
 Copyright, 68
 Corpus, 11, 18, 50, 55, 61, 88, 121, 154
 manual analysis, 72, 88
 Correction, 100, 140
 Cost, 10, 136, 159
 life-cycle, 136, 152
 Covid pandemic, 66, 70, 155, 170
 Covid reporter, 169
 Creative Commons, 18, 68
 Crowdworkers, 61, 62, 113, 116
 Culture, 144

D

Data augmentation, 64, 70
 Data availability, 160, 167
 Data contamination, 104, 126, 139
 Data interpretation, 5, 26, 28, 29, 33, 46, 52
 abstraction, 33, 41
 importance, 34, 35
 interpretation, 33
 linkage, 33, 36, 52
 Data leakage, 148
 Data protection, 104
 Data science, 46
 Data sheet, 102
 Data-to-text (Data2Text), 3, 5, 10, 20, 22, 25, 26, 30, 33, 66, 160
 Decision support, 17, 83, 96, 176
 Deep learning, 9, 11, 20, 57
 Definite descriptions, 39
 Demographics, 114
 Depression, 146
 Doctors, 6, 14, 29, 61, 84, 91, 96, 103, 110, 135, 144, 163, 172, 173, 176
 Document plan, 36

Document planning, 5, 26, 28, 29, 35, 40, 47, 52
 Document structure, 35
 Domain, 164
 experts, 3, 14, 26, 34, 38, 45, 61, 89, 108, 110, 114, 149
 knowledge, 2, 10, 26, 34, 113, 152
 shift, 15, 66, 67, 70, 147
 DrivingFeedback, 28, 31, 33, 35, 37, 39, 40

E

Ecological validity, 103, 134
 Edge cases, 15, 45, 90, 115, 138
 Edit distance, 122
 Elision, 40
 Emotion, 146, 149, 178
 Encoder-decoder, 57
 Engineers, 22
 English, 42
 Error analysis, 8
 Errors, 9, 79, 161, 167, 174
 critical, 75
 reporting, 100
 severity, 108
 Esperanto, 42
 Ethics, 14, 18, 114, 120, 135
 E2E, 62, 76
 EU Artificial Intelligence Act, 70
 Evaluation, 112, 151
 granularity, 131, 132
 Examples, 63, 65
 Experimental design, 99, 110, 119–121, 124, 140
 Experimental execution, 100, 119, 139, 140
 Exploratory Data Analysis, 117

F

Few-shot, 65
 Fiction generation, 3, 75
 Fidelity, 74
 Fine-tuning, 8, 52, 57, 58, 126, 129, 155
 Flesch-Kincaid, 123
 Fluency, 11, 12, 73
 FoG, 20, 22
 Followup, 100
 French, 42
 Friedman test, 117

G

Galician, 42
 GEM, 140

GEMBA-MQM, 129
 Gemini, 1, 130
 Gender stereotypes, 18
 Generalisation, 51
 Genre, 10, 26, 90, 164
 German, 42
 Google Books, 51
 GPT, 58, 66, 124, 129, 146–148
 Grammar, 41
 Graphics, 13, 16, 20, 71, 82, 92, 96, 169
 information, 72
 Guidelines, 81, 119

H

Hackers, 138
 Hallucinations, 74, 145, 178
 extrinsic, 75
 intrinsic, 75
 Health, 16, 17, 29, 38, 63, 64, 66, 74, 75, 78,
 80, 139, 161, 172, 179
 Historical comparison, 15, 134
 Huggingface, 23, 58, 60, 70, 141
 Human authors, 108
 Human checking, 80
 Human-computer interaction (HCI), 2, 3, 8, 92
 Human evaluation, 14, 22
 Human-in-loop, 12, 13, 49, 77, 149
 Hypothesis testing, 14, 97, 116, 151

I

IBM Watson, 16, 66, 92, 162, 163, 179
 Image captioning, 20
 Image generation, 3
 Impact, 9, 14, 101, 103, 134, 172
 evaluation, 94, 105, 110, 130, 134
 Individual variability, 37
 Inferences, 40
 Inflect (Python package), 43, 46
 Insights, 33, 35, 37, 40, 45, 76, 116
 Institutional review board, 120
 Instruction tuning, 58, 60, 65
 Integration, 80, 136
 Intellectual property, 18, 68
 Inter-annotator agreement, 118, 131
 Interquartile Range (IQR), 117
 Inuit, 156
 Investors, 137
 IT, 152

J

Jinja, 26, 43, 45

Job losses, 18, 137
 Journalism, 16, 18, 26, 62, 75, 78, 81, 92, 159,
 161, 164, 179
 automatic, 164
 data, 164

K

Kaggle, 61, 62
 Kappa, 118
 Cohen's, 118
 Fleiss, 118
 statistic, 119
 Keras, 60
 Knowledge distillation, 53
 Krippendorff's alpha, 118
 Kruskal-Wallis test, 117

L

Language change, 51
 Language functions, 43, 46
 Language models, 8, 12, 60, 156
 Large language models, 11, 14, 18, 43, 49, 58,
 63, 74, 104, 147, 155, 156, 165, 166,
 168, 171, 178
 Latin square design, 116
 Lawsuits, 69, 138, 156
 Legal, 66, 68, 70, 74, 75, 91, 121, 147, 156
 Lexical choice, 37, 39, 45–47, 55
 Lexical variation, 37, 44
 Liability, 10, 149
 Libraries, 10, 20, 42
 Lies, 19, 87
 Likert scale, 108, 117
 Linear interpolation, 32
 Linear regression, 32
 Linguistics, 2, 10, 35, 37, 39, 41
 Linguists, 3, 53
 Literacy, 74
 Logistics, 178
 Long short-term memory (LSTM), 57
 Loss-leader, 137

M

Machine translation, 3, 19, 81, 93, 98, 124, 132,
 140
 Magnitude estimation, 108
 Mail merge, 43
 Maintenance, 10, 15, 21, 45, 66, 136, 143, 147,
 152, 157, 161
 Managers, 91, 135
 Mandarin, 42
 Mann-Whitney, 117

Maps, 82, 85
 Material, 99, 104, 114, 126
 Meaning representation, 52
 Mechanical Turk, 61, 113
 MedPaLM, 67, 178
 Messages, 33, 37, 40
 Metrics, 8, 14, 20, 121, 125, 126, 128
 reference-based, 122, 126, 128, 129
 referenceless, 123, 126
 validation, 122, 130, 132, 140, 152
 Microplanning, 5, 26, 28, 29, 37, 41, 43, 47
 Microsoft Excel, 45
 Microsoft Word, 45
 Misinformation, 18
 Mockups, 87
 Models, 11, 60, 63, 129, 156
 bigram, 51, 56
 fine-tuned, 49, 62
 foundation, 11, 49, 57
 interpretable, 34, 38
 Markov, 56
 ngram, 51, 56, 128
 prompted, 49, 53, 63, 67, 68, 155, 160
 trained, 49, 50
 Monitoring, 150
 Morphology, 41, 46
 Morphophonology, 42
 MQM, 108, 112, 129
 Multi-modal, 84, 170

N

Narrative, 25, 35
 Narrative Science, 20
 National Health Service, 121
 Natural language understanding, 2
 Negative results, 95
 News
 fake, 18, 168
 financial, 167
 general, 166
 sports, 166
 Nltk, 56, 70
 Noise detection, 31, 34, 46
 Non-native speakers, 40, 74, 78
 Note Generator, 6, 11, 58, 61, 86, 108, 134, 135, 164, 171, 174
 Null hypothesis, 98
 Numeracy, 176
 Numpy, 46
 Nurses, 29

O

Oil industry, 3, 22, 31

Omissions, 76, 178
 OpenCCG, 43
 Orthography, 42, 46
 Outlier, 113, 117, 120
 Over-generation, 43

P

Pain point, 163
 PaLM, 58
 Parameters, 57
 Paraphrase, 64
 Patient information, 17, 175
 Pattern detection, 31
 Peer review, 140
 Personas, 87
 Persuasion, 18
 Pilot experiments, 116, 119
 Pipeline, 10, 12, 26, 30, 33, 35, 37, 44, 45, 60
 Post-editing, 7, 9, 71, 80, 92, 161, 165, 174
 automatic, 81
 time, 77, 110, 135
 PowerBI, 169
 Pragmatics, 39, 75
 Pre-registration, 99
 Privacy, 78, 135
 Probability, 176
 Procedure, 99, 115, 126
 Production, 14, 88, 168
 Prolific, 61, 113
 Prompt, 11, 53, 58, 65, 104, 125, 129
 engineering, 11, 58
 Pronouns, 39, 60
 Provenance, 68
 Psycholinguistics, 36
 Psychologists, 96, 103
 Psychology, 2, 3
 P value
 one-tailed, 98
 two-tailed, 98
 Pyrealb, 42

Q

Qualitative error analysis, 118, 128
 Quality assurance, 151, 157
 Quality criteria, 13, 71, 72, 77, 79, 92, 93, 97, 101, 112, 116, 121, 123, 125, 131, 132, 139

R

Radar, 166

Randomised controlled trial, 134
 Random seeds, 102
 Ranking, 14, 105, 112, 114, 117, 140
 Rating, 6, 14, 105, 112, 114, 117, 140
 Readability, 11, 13, 71, 73, 93
 Recurrent neural networks, 57
 Reddit, 64, 179
 Red teams, 149, 151
 Reference target, 39
 Reference texts, 14, 121, 122, 126, 128, 152
 Referring expressions, 22, 37, 39, 45–47
 Regression testing, 152
 Regulator, 68, 70, 91, 150, 152, 167, 169
 Reinforcement Learning from Human
 Feedback (RLHF), 58, 60, 65
 Replication, 101, 119, 122, 124, 139, 140
 Reporting, 17, 173
 Representativeness, 62, 104, 114
 ReproHum, 100, 140
 Requirements, 13, 25, 71, 93, 112, 155, 163
 acquisition, 13, 92
 analysis, 71, 85
 non-functional, 77
 Research ethics committee, 120
 Research questions, 99, 101, 112, 124, 136
 Retraction, 140
 Retrieval-augmented generation, 60, 67
 Return on investment, 16, 135, 138, 159
 Rhetorical Structure Theory, 36
 Risk, 38, 91, 137, 143, 175, 176
 Robustness, 15, 136
 Rosaenlg, 43
 ROUGE, 20, 122, 129
 Rule-based NLG, 5, 10, 19, 25, 53, 90,
 132, 144, 147, 151, 152, 154,
 165, 166

S

SaferDriver, 28, 85, 86
 Safety, 15, 17, 63, 68, 80, 138, 139, 143, 155,
 157, 161, 177, 178
 Scalability, 10, 16, 17, 159, 160, 172, 173
 Scenarios, 114, 126
 Schemas, 36
 Scikit-learn, 55, 60
 Scipy, 33, 46, 70
 Scripts, 36, 43, 45
 Security, 77, 78
 Self-driving cars, 137
 Semantics, 37, 75
 lexical semantics, 37
 Semi-structured interviews, 87
 Sequence-to-sequence, 57

Signal analysis, 5, 26, 28, 30, 33, 46, 52
 Simplenlg, 30, 41, 42, 47
 Siri, 145
 Society, 18
 Software engineering, 2, 3, 10, 15, 71, 92, 152,
 157
 Spam, 63
 Sports-writing, 44
 Stakeholders, 13, 87, 91, 93, 96, 101, 103, 139
 State-of-art, 101, 112, 125, 139
 Statistical, 140
 analyses, 140
 hypothesis testing, 98
 significance, 95, 99, 102, 117, 127, 139
 tests, 98, 117
 Stochastic, 15, 152
 Stock markets, 167
 STOP, 94, 97, 98, 112, 134, 135, 175
 Stress, 146
 Structured survey, 132
 Study type, 99, 112, 126
 Subjects, 99, 102, 113, 116, 117, 119, 120, 132,
 135
 Sublanguage, 164
 Summarisation, 6, 16, 60, 170, 174
 abstractive, 170, 171
 extractive, 170
 SumTime, 37, 38, 55, 81, 105, 112, 114, 153,
 164
 Support, 152
 Surface realisation, 5, 26, 29, 30, 41, 156
 Surface realisers, 47
 Survey design, 140
 Syntax, 37, 41, 46
 Synthetic data, 64, 70, 104

T

Tableau, 169
 Target texts, 72, 88
 Task-based evaluation, 9, 14, 20, 109, 112, 114,
 121, 134
 Templates, 26, 43, 47, 168
 Terminology, 4, 38, 78, 164
 Test cases, 15, 151, 153
 Test data, 104, 139
 Testing, 10, 12, 15, 21, 25, 93, 143, 149, 151,
 155–157
 Test set, 126
 Text-to-text, 3, 6, 10
 T5, 11, 58
 Thematic analysis, 118
 Third parties, 135
 Thomson-Reuters, 168

Time series, 31
Toxic content, 64
Toxic language, 144, 149
Training, 116, 119, 150
Training data, 10, 11, 51, 52, 58, 60, 104, 161, 163
Transactions of the ACL, 100
Transformers, 57
Transparency, 10
TrueSkill, 107
Trust, 163, 177
T-test, 117

U
Under-represented communities, 18
Under-resourced languages, 18
Usability, 3
Use cases, 10, 13, 16, 26, 71, 76, 78, 80, 82, 93, 144, 155, 159
User interface, 81, 100, 102, 110, 116, 120, 135
User needs, 15
User studies, 72, 85
Utility, 6, 9, 14, 16, 76, 103, 134, 137

V
Vagueness, 47
Variation, 78, 152, 167, 168
Visualisations, 13, 16, 82, 85
Volume, 159

W
Weather forecasts, 3, 18, 19, 22, 37, 52, 55, 60, 61, 64, 76, 78, 80, 81, 83, 91, 114, 139, 154, 155, 164, 170
Web-based experiments, 116
Welsh, 165
Wikipedia, 50, 56, 68
Wizard of Oz, 88
WMT, 112, 131, 140
Workflow, 13, 71, 80, 92, 93, 110, 135, 163, 178
Worst-case, 71, 79, 80, 127, 138, 143, 151, 152
Writing assistants, 3

Z
Zero-shot, 65