



Ask and distract

**Data-driven methods for the automatic generation of multiple-choice reading
comprehension questions from Swedish texts**

DMYTRO KALPAKCHI

Doctoral Thesis in Speech and Music Communication
Stockholm, Sweden, 2023

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
Division of Speech, Music and Hearing
SE-10044 Stockholm
Sweden

TRITA-EECS-AVL-2023:56
ISBN 978-91-8040-661-1

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av Teknologie doktorexamen i tal- och musikkommunikation tisdagen den 17 oktober 2023 klockan 14.00 i Sal F3, Lindstedtsvägen 26, Kungliga Tekniska Högskolan, Stockholm.

© Dmytro Kalpakchi, August 7, 2023

Tryck: Universitetsservice US AB

Abstract

Multiple choice questions (MCQs) are widely used for summative assessment in many different subjects. The tasks in this format are particularly appealing because they can be graded swiftly and automatically. However, the process of creating MCQs is far from swift or automatic and requires a lot of expertise both in the specific subject and also in test construction.

This thesis focuses on exploring methods for the automatic MCQ generation for assessing the reading comprehension abilities of second-language learners of Swedish. We lay the foundations for the MCQ generation research for Swedish by collecting two datasets of reading comprehension MCQs, and designing and developing methods for generating the whole MCQs or its parts. An important contribution is the design of methods for the automatic and human evaluation of the generated MCQs were also designed and applied in practice.

The best currently available method for generating MCQs (as of June 2023) for assessing reading comprehension in Swedish is ChatGPT (although still only around 60% of generated MCQs were judged acceptable). However, ChatGPT is neither open-source, nor free. The best open-source and free-to-use method is the fine-tuned version of SweCTRL-Mini, a foundational model developed as a part of this thesis. Nevertheless, all explored methods are far from being useful in practice but the reported results provide a good starting point for future research.

Keywords: multiple choice questions, question generation, distractor generation, reading comprehension, second-language learners, L2 learning, Natural Language Generation

Sammanfattning

Flervalsfrågor används ofta för summativ bedömning i många olika ämnen. Flervalsfrågor är tilltalande eftersom de kan bedömas snabbt och automatiskt. Att skapa flervalsfrågor manuellt går dock långt ifrån snabbt, utan är en process som kräver mycket expertis inom det specifika ämnet och även inom provkonstruktion.

Denna avhandling fokuserar på att utforska metoder för automatisk generering av flervalsfrågor för bedömning av läsförståelse hos andraspråksinlärare av svenska. Vi lägger grunden för forskning om generering av flervalsfrågor för svenska genom att samla in två datamängder bestående av flervalsfrågor som testar just läsförståelse, och genom att utforma och utveckla metoder för att generera hela eller delar av flervalsfrågor. Ett viktigt bidrag är de metoder för automatisk och mänsklig utvärdering av genererade flervalsfrågor som har utvecklats och tillämpats i praktiken.

Den bästa för närvarande tillgängliga metoden (i juni 2023) för att generera flervalsfrågor som testar läsförståelse på svenska är ChatGPT (dock bedömdes endast cirka 60% av de genererade flervalsfrågorna som acceptabla). ChatGPT har dock varken öppen källkod eller är gratis. Den bästa metoden med öppen källkod som är också gratis är den finjusterade versionen av SweCTRL-Mini, en “foundational model” som utvecklats som en del av denna avhandling. Alla utforskade metoder är dock långt ifrån användbara i praktiken, men de rapporterade resultaten ger en bra utgångspunkt för framtida forskning.

List of Papers

Contributions by the author included in the thesis.

- I. ***Quinductor: A multilingual data-driven method for generating reading-comprehension questions using Universal Dependencies***
Dmytro Kalpakchi, Johan Boye
Natural Language Engineering (2023), 1-39. doi:10.1017/S1351324923000037
- II. ***Automatically generating question-answer pairs for assessing basic reading comprehension in Swedish***
Dmytro Kalpakchi, Johan Boye
The Ninth Swedish Language Technology Conference (SLTC 2022), Stockholm, Sweden.
- III. ***Minor changes make a difference: a case study on the consistency of UD-based dependency parsers***
Dmytro Kalpakchi, Johan Boye
In Proceedings of the Fifth Workshop on Universal Dependencies (UDW, SyntaxFest 2021), pages 96–108, Sofia, Bulgaria.
- IV. ***BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset***
Dmytro Kalpakchi, Johan Boye
In Proceedings of the 14th International Conference on Natural Language Generation (INLG 2021), pages 387–403, Aberdeen, Scotland, UK.
- V. ***Quasi: a synthetic Question-Answering dataset in Swedish using GPT-3 and zero-shot learning***
Dmytro Kalpakchi, Johan Boye
In Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa 2023) (pp. 477-491), Tórshavn, Faroe Islands.
- VI. ***SweCTRL-Mini: a data-transparent Transformer-based large language model for controllable text generation in Swedish***
Dmytro Kalpakchi, Johan Boye
arXiv preprint arXiv:2304.13994
A non-peer reviewed manuscript

VII. ***Generation and Evaluation of Multiple-choice Reading Comprehension Questions for Swedish***

Dmytro Kalpakchi, Johan Boye

urn:nbn:se:kth:diva-329400

A non-peer reviewed manuscript

VIII. ***UDon2: a library for manipulating Universal Dependencies trees***

Dmytro Kalpakchi, Johan Boye

In Proceedings of the Fourth Workshop on Universal Dependencies (UDW, COLING 2020), pages 120–125, Barcelona, Spain (Online).

IX. ***Textinator: an Internationalized Tool for Annotation and Human Evaluation in Natural Language Processing and Generation***

Dmytro Kalpakchi, Johan Boye

In Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC 2022), pages 856–866, Marseille, France.

Other contributions by the author not included in the thesis.

- ***SpaceRefNet: a neural approach to spatial reference resolution in a real city environment***

Dmytro Kalpakchi, Johan Boye

In Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue (SIGDial 2019), pages 422–431, Stockholm, Sweden. Association for Computational Linguistics.

- ***EMBRACE: Evaluation and Modifications for Boosting RACE***

Mariia Zyrianova, **Dmytro Kalpakchi**, and Johan Boye

arXiv preprint arXiv:2305.08433

A non-peer reviewed manuscript

- ***Collecting Visually-Grounded Dialogue with A Game Of Sorts***

Bram Willemsen, **Dmytro Kalpakchi**, and Gabriel Skantze

In Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC 2022), pages 2257–2268, Marseille, France. European Language Resources Association.

Acknowledgement

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Acronyms

List of commonly used acronyms:

CEFR	Common European Frame of Reference
IAA	Inter-annotator agreement
MCQ	Multiple-choice question
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
QG	Question Generation
SFI	Swedish for Immigrants

Contents

List of Papers	iii
Acknowledgement	v
Acronyms	vii
Contents	1
1 Introduction	3
1.1 Notation	4
2 Background: Test Item Formats	7
3 Background: Natural Language Generation	13
3.1 Rule-based approaches	15
3.2 Data-driven approaches	17
3.3 Human evaluation	23
3.4 Automatic evaluation	24
4 State-of-the-art Overview	35
5 Summary of the Included Papers	41
6 Discussion and Conclusions	53
6.1 Limitations	53
6.2 Why not translate automatically?	54
6.3 Societal impact	54
6.4 Contributions towards Open science	55
6.5 Contributions towards Sustainability	56
6.6 Future work	56
6.7 Conclusions	57
References	59

Chapter 1

Introduction

In what way can we assess how well a person is able to read? One way would be to let them read a couple of texts, then discuss some questions about these texts with them and assign a grade based on this discussion. The problem with this approach is that it does not scale well. As the number of second-language learners around the world steadily increases due to increased mobility, it becomes harder for teachers to use this methodology without sacrificing quality.

In Sweden in particular, the number of migrants has grown substantially in the latest decade and is projected to grow steadily after 2030, as reported by Statistics Sweden.¹ For these migrants, learning Swedish is an essential part of integrating into the Swedish society. Indeed, the demand for learning Swedish is high, as demonstrated, for instance, by the 2022 Duolingo Language Report² showing that Sweden is the only country in Europe (and one of the very few in the world) where the official language, Swedish, is the most popular among Duolingo learners in the country.

In what way can we design a scalable assessment of a person's reading abilities, then? The preferred way at the time of writing is to use reading comprehension tests, for instance, IELTS³ or TOEFL⁴ for English, or Tisus⁵ for Swedish. These meticulously designed tests provide learners with a number of carefully chosen texts, together with associated tasks (also called *test items*). After taking the test, the learners are then assigned one of six possible skill levels (from A1 to C2) according to the widely adopted Common European Frame of Reference (CEFR; Council of Europe (2001)).

Attaining each CEFR level requires the students to read texts of certain com-

¹<https://www.scb.se/en/finding-statistics/statistics-by-subject-area/population/population-projections/population-projections/pong/tables-and-graphs/immigration-and-emigration-by-sex-and-country-of-birth-and-projection/>

²<https://blog.duolingo.com/2022-duolingo-language-report/>

³<https://www.ielts.org/>

⁴<https://www.ets.org/toefl.html>

⁵<https://www.su.se/tisus/>

plexity and complete a number of test items in various formats based on these texts. PISA 2025 Foreign Language Assessment Framework (OECD, 2021, Annex Table 4.A.1) lists a number of test item formats (further discussed in Chapter 2), of which only one, namely the multiple-choice format, is used for assessment at *every* CEFR level. Broadly speaking, test items in this format consist of a *stem* (which formulates the task), and a number of *alternatives* (which provide possible solutions to that task). A test taker then has to choose a (specified or unspecified) number of alternatives that they deem correct. The tasks in multiple-choice format have a very appealing advantage of being graded swiftly, automatically, and objectively (since grading only requires matching the alternatives selected by a student with the key sheet).

The focus of this thesis is on *conventional multiple-choice format* with four alternatives, of which only one is correct (further discussed in Chapter 2). The primary goal of this work is to develop methods capable of automatically generating tasks in multiple-choice format for assessing the reading comprehension abilities of second-language learners of Swedish. The aim of the developed methods is to shorten the test design time, and **NOT** to automate away teachers or test constructors.

The rest of the thesis is structured as follows:

- Chapter 2 supplies the necessary background information on formats of test items for reading comprehension, and outlines the specifics of each test item format, and their relation to conventional MCQs;
- Chapter 3 gives a brief overview of the Natural Language Generation field, and provides pointers for further reading;
- Chapter 4 gives a brief overview of the state-of-the-art for automatic generation of reading comprehension MCQs;
- Chapter 5 re-iterates the problem definition, and provides one-page summaries for the included papers listing contributions and relating them to the problem definition;
- Chapter 6 discusses limitations and potential societal impact of this thesis, as well as concerns and contributions towards the sustainable development goals, and open science.

1.1 Notation

Before we dive into the rest of the thesis, we define the following notations adopted throughout this thesis *whenever using mathematical expressions*:

- text in a **monospace** font, e.g. `c`, denotes variables containing strings;
- lowercase letters in *italics*, e.g., *h*, denote scalars;

- uppercase letters in *italics*, e.g., H , denote ordered sets;
- lowercase letters in **bold**, e.g. \mathbf{h} , denote vectors, and a subscript i , e.g. \mathbf{h}_i , denotes i -th element of the vector \mathbf{h} ;
- uppercase letters in **bold**, e.g. \mathbf{A} , denote matrices, and a subscript i, j , e.g. $\mathbf{A}_{i,j}$, denotes the element in the row i and the column j of the matrix \mathbf{A} , which is also referred to as ij -element of \mathbf{A} ;
- for distinguishing between different vectors denoted by the same letter, we adopt a parenthetical superscript, i.e. $\mathbf{h}^{(1)}$ or $\mathbf{h}^{(2)}$, whereas the fourth element of each of these vectors would be denoted as $\mathbf{h}_4^{(1)}$ and $\mathbf{h}_4^{(2)}$, respectively;
- the same as above applies to matrices, where $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ are different matrices, and $\mathbf{A}_{i,j}^{(1)}$ and $\mathbf{A}_{i,j}^{(2)}$ denote the ij -element of $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ respectively;
- in both cases parenthetical indices should not necessarily be numbers, but could also include letters e.g., $\mathbf{W}^{(x)}$ or $\mathbf{h}^{(t)}$.

Note that these notations are not necessarily consistent with the notation used in each of the included articles.

Chapter 2

Background: Test Item Formats

Any test is composed of a number of tasks, usually referred to as *test items*. Haladyna (2004) defines a *test item* as “a statement that elicits a test taker response”. Observe that this “statement” is not necessarily declarative, and is in fact rather often interrogative.

Haladyna (2004) emphasizes that the fundamental difference in formats for any test items is whether the answer is created or selected by the test taker. There are three broad categories of test item formats (Haladyna, 2004; OECD, 2021), all of which are applicable when testing reading comprehension:

- **constructed response**, when the test takers are asked to provide a response in their own words;
- **multiple choice**, when the test takers are prompted to select a response from a number of alternatives for each formulation;
- **matching**, when the test takers are presented with multiple formulations and multiple response options, and they have to pair each formulation with an option.

This work is focused *exclusively* on test items in the multiple choice format, although in this section we briefly describe the other formats as well. A better understanding of the between-format differences should help the reader later in understanding the limitations when applying the methods developed in this thesis to formats other than multiple-choice.

In the context of assessing reading skills, a major part of a test item in any of these formats is a text. Carefully choosing and adapting texts is a complex task in itself, which is, however, not a part of this work. For the purposes of this thesis we assume the sufficient texts are *given in advance*. The interested reader is referred to (OECD, 2019, Chapter 2) for more details on the types of texts and the ways to choose them.

A WOMAN LEARNS TO READ

Ndugu Rukia Okashi is a 53-year old farmer living in Arusha, Tanzania. She grows maize, beans, and vegetables, has seven children, and she learned to read about ten years ago. She says:

“There is a great difference in my present situation when compared with the old days. A lot of changes have taken place. When I had to sign papers and documents, I could only use the thumb-print and I never knew exactly what I was signing. So I was sometimes cheated. Now that I can read and write no one can ask me to sign just blindly. I first have to ask myself, and it is only after I am satisfied that I agree to sign. If I don’t agree with the contents of the documents, I just don’t sign.

Now that I can read, I know which food is good to make me strong, which keep me well, and so on. I now can give my children a balanced diet.

In the old days, when one walked through the streets one couldn’t read any signs. You may come across a ‘Danger’ signboard but you continue to walk ahead until someone shouts, ‘Mama, mama, mama, mama, stop!’ But these days, I can read all the sign-posts such as ‘Don’t pass here; Keep out.’ In traveling also, I used to ask the driver to let me get off at a certain place, but sometimes the driver would take you much further beyond your destination. If such an incident occurs now, I shout and protest.

So now I feel great and self-confident. Now I can refuse or disagree where formerly I used to be the victim of other people because I was illiterate.”

Figure 2.1: A text from the IEA 1991 Reading Literacy study, for pupils of the 9th grade, Session I, questions 37 - 41

To exemplify different test item formats, we again assume a text is given, and borrow the one in Figure 2.1 along with its related test items, from the Reading Literacy Study (RLS) conducted during 1990/1991 by the International Association for the Evaluation of Educational Achievement (IEA). Both the text and the corresponding test items are publicly available in the Attachment A-1 of the report by Binkley and Rust (1994). For clarity, the test items taken from the IEA RLS will be numbered using the arabic numbers from the IEA RLS (e.g., 37, 39, 40), whereas the test items constructed for the presentation purposes will be prefixed with the capital letter X (e.g., X-1, X-2, X-3). **All test items in the sections below are based on the text in Figure 2.1.**

Multiple choice

Haladyna (2004); Haladyna et al. (2002) define different subformats within the multiple-choice format. The most common subformat is called *conventional mul-*

multiple choice, which consists of the three parts:

- *a stem* inquiring about some information, also defined as “the stimulus for the response” by Haladyna (2004);
- *the key*, which is the correct answer providing the sought information;
- a number of wrong, but plausible answers, called *distractors*.

Depending on the form of a stem, Haladyna (2004) further distinguishes *the question format* (with an interrogative stem, as in the test item 39), *the incomplete stem format* (with the stem being a partial sentence, as in the test item 40), or *the best answer format* (with the stem asking to find the best match among the given alternatives, even if all of the them are correct in their own right, as in the test item 37).

39. What advantage does she see in learning to read signs?

- A. She can avoid trouble.
- B. She can improve her income.
- C. She is not so likely to be cheated.
- D. She doesn't make her children so ashamed.

40. Mrs. Okashi often protests loudly if she is

- A. charged too much by drivers.
- B. taken too far by drivers.
- C. mistakenly gets into danger.
- D. misled by public signs.

37. Which of these phrases best expresses the underlying theme of this passage?

- A. The benefits of becoming literate.
- B. The way in which one person became literate.
- C. The problems of being an illiterate Tanzanian farmer.
- D. The difficulties of coping in a literate world.

Another type of stem for the conventional multiple-choice format is the one including blank(s), also called *blank-type items*, as in the test item X-1.

X-1. Mrs. Okashi often protests _____ if she is taken too far by drivers.

- A. violently
- B. silently
- C. loudly
- D. quietly

While all test items above include four alternatives (A, B, C, and D), the “optimal” number of alternatives is unclear Haladyna and Downing (1993). However,

there are also separate multiple-choice format types that include only two alternatives on purpose. One such type is called *alternate choice format*, which is a conventional multiple-choice format with two comparable options (as in the test item X-2).

- X-2. Mrs. Okashi often protests loudly if she is
- A. taken too close by drivers.
 - B. taken too far by drivers.

Another very similar type is the *true-false format* which provides a declarative statement and asks the test taker whether it is true or false (as in the test item X-3). While the exact formulation could differ (e.g., right or wrong, yes or no), the focus should still be on identifying the veracity of the statement.

- X-3. Identify whether the following statements are true or false.
- Mrs. Okashi is a teacher from Tanzania that could not read.
 - Mrs. Okashi often protests loudly if she is taken too far by drivers.

There are also a number of *complex multiple-choice format* containing, for instance, multiple true-false items sharing the same premise, or dependent item sets. These formats constitute either variations or combinations of the subformats outlined above (albeit with some peculiarities). The interested reader is referred to Haladyna (2004); Haladyna et al. (2002) for further information on the matter.

In this thesis we worked exclusively with conventional multiple choice format with interrogative stems. We focused on the test items with four alternatives, of which only one is correct.

Matching

Haladyna (2004); Haladyna et al. (2002) view the *matching format* as a part of multiple-choice format, whereas OECD (2021) consider it a different format. From the perspective of their automatic generation, the matching format is rather different, which is why it is highlighted separately.

Test items in the matching format consist of a set of stems and a set of options. The test takers are then asked to find one matching option for each stem, as exemplified in the test item X-4.

- X-4. Match each characteristic of Mrs. Okashi in the left column with the numeric value in the right column.
- | | |
|--|-------|
| 1. Mrs. Okashi's age | A. 6 |
| 2. Number of Mrs. Okashi's children | B. 10 |
| 3. For how many years Mrs. Okashi can read | C. 53 |
| | D. 7 |

In this thesis we did **NOT** consider test items in the matching format, and neither datasets nor methods developed in this thesis are expected to work for the MF (not least because the stems here are not interrogative).

Constructed response

Question 41, presented below, is a typical constructed response (CR) test item. In this test item, the formulation contains a question and instructions on how to proceed while answering the task. The answer must be produced (constructed) by the test takers themselves.

41. What do you think would be the disadvantages in your country for an adult who could not read or write?

Write your answer on the lines below. Make sure you give enough information to make your answer clear. You may want to use references from the passage to explain your ideas.

In this thesis we did **NOT** consider CR test items, although many stems generated for the test items in multiple-choice format could be re-used for CR. However, some stems, e.g., those requiring explicit selection from a list of alternatives, will most definitely not work.

Chapter 3

Background: Natural Language Generation

Natural Language Processing (NLP) is the field of Artificial Intelligence that studies methods enabling computers to understand and produce natural language, e.g., in English or Swedish. This thesis focuses on methods for producing language, which is the subject of interest for the subfield of NLP called Natural Language Generation (NLG). Indeed, the focus of this work is on using NLG methods for the automatic generation of reading comprehension test items in the multiple-choice format. The aim of this chapter is to give a brief introduction to the NLG techniques relevant to this thesis and provide pointers for further reading to interested readers.

Gatt and Krahmer (2018) divide NLG tasks into the following three types:

- **text-to-text** generation, when both input and output are texts in natural language, for instance, as in text summarization (e.g., surveyed by El-Kassas et al. (2021)), or paraphrase generation (e.g., surveyed by Zhou and Bhat (2021));
- **vision-to-text** generation, when the input is a visual stimulus (e.g., image or video) and the output is a text in natural language, for instance, as in image captioning (e.g., surveyed by Stefanini et al. (2022)), or generating referring expressions for specified parts of images (e.g., surveyed by Krahmer and Van Deemter (2012));
- **data-to-text** generation, when the input constitutes neither linguistic nor visual data, while the output is still a text in natural language, for instance, as in automated journalism (e.g., (Leppänen et al., 2017)), or providing summaries of medical measurements (e.g., (Banaee et al., 2013)).

Recall that the focus of this work is on creating methods for the automatic generation of reading comprehension test items in the multiple-choice format. This

is a clear instance of text-to-text generation, which will be the main focus for the remainder of this chapter.

When creating any text-to-text NLG system, one is confronted with two major design decisions. The first one is how to view the problem: decomposing it into smaller sub-problems, attempting to solve each sub-problem separately, and then combining the solutions (a *modular* approach), or attempting to solve the whole problem at once (an *end-to-end* approach). The second design decision is which algorithms¹ to use for text generation: those based on rules (later referred to as *rule-based*), or those learning from data (later referred to as *data-driven*). The current predominant approach in the field is to design data-driven, where both the input and output constitute raw text, and the processing in-between is completely learned from data using Machine Learning (ML) techniques. The aforementioned dominance can be attributed to substantial performance improvements across different NLG tasks achieved by using one particular kind of ML algorithm, called *artificial neural networks* (or simply neural networks). This class of algorithms is of core interest to this thesis, and is introduced further in Section 3.2.

Despite the substantial performance gains over both rule-based algorithms and more traditional ML algorithms, neural networks have a major problem of being *uninterpretable*. Currently, researchers do not know how neural networks solve the text-to-text NLG problems exactly, although there are research efforts in this direction, e.g., (Saha et al., 2022; Shi et al., 2020; Wiseman et al., 2018).

Another substantial problem is that the best performance has traditionally been achieved by letting neural networks learn from manually annotated data (so-called *supervised learning*). The required data volume is large and, what is more problematic, the exact amount of data necessary for each given problem is currently impossible to estimate. Annotated datasets for the Swedish language are scarce compared to English, and annotating new data is very costly both in terms of time and money. In this thesis, we explored two ways to remedy this problem: (1) making use of *unannotated* data by using pre-trained (sometimes called foundational) models, and (2) designing methods that require smaller volumes of annotated data.

The first alternative above is currently very popular largely due to the success of *Transformers* (Vaswani et al., 2017), a particular kind of neural network briefly introduced in Section 3.2. Transformers are trained using vast quantities of unannotated raw text on artificial tasks, such as predicting the next word in a sequence (used by, for example, GPT-3 (Brown et al., 2020)), or attempting to restore words in a corrupted sequence (used in the training of BERT (Devlin et al., 2019)). The data for such kinds of tasks can be “annotated” automatically, without humans in the loop, marking a shift from supervised to *self-supervised* learning. With this approach, data quantity is no longer a problem, as long as there is raw text in digital format, which is perfect for training large neural net-

¹Algorithms based on AI planning have also been used by the NLG researchers, but are not discussed in this thesis, as this thesis does not build on them in any capacity.

works on text (called *large language models*). The training of these models could take months using vast quantities of high-performing hardware, and the hope is that they attain some kind of knowledge about the structure and semantics of the language(s) present in the training data. Once they are trained, one could attempt to use them for any language-related tasks, in two ways:

- **use the model as-is**, either by *zero-shot learning*, i.e. providing only a description (or a template) of the task and then asking the model to complete an example (for instance, “Write a summary of the text below” followed by the actual text), or by *few-shot learning*, i.e. providing a couple of correct examples in addition to the task description;
- **train the model further on your specific task**, which requires potentially hundreds or thousands of examples in order to train the model for the specific task at hand, a process called *fine-tuning*.

In this thesis we explored both ways of building on large language models.

The second way to avoid the need for large quantities of labeled data is to design methods that require substantially smaller amounts of such data. This approach is less popular, perhaps because it is currently less clear how to reach a good level of performance while using substantially smaller training datasets. In this thesis, we have explored the use of templates for generating question-answer pairs, an approach traditionally requiring a lot of manual labor to craft these templates. To remedy that, we proposed learning these templates (and rules for when they can be applied) from relatively small Question Answering datasets (as described in Paper I). To ensure that the induced templates are applicable beyond the texts and question-answer pairs seen during training, we relied on regularities of syntactic structures using *Universal Dependencies* as the intermediate representation (briefly introduced in section 3.1).

The remainder of this chapter briefly introduces the core NLG concepts, which have been mentioned up until this point and are relevant for understanding the papers included in the thesis. The aim of this chapter is to introduce terms that are widely used by the NLP/NLG community but might not necessarily be as familiar to researchers from the adjacent fields. Additionally, some papers also use terms and methods that are used relatively rarely even within the NLP/NLG community. In such cases, these papers contain the necessary background information, which will not be repeated in this chapter.

3.1 Rule-based approaches

One of the earliest approaches to text-to-text NLG was based on rules, which could be either hand-written or induced from data. One task for which such approaches were fruitful is text simplification (TS). For instance, Suter et al. (2016) applied hand-written rules for TS in German, and Chandrasekar and Srinivas

(1997) learned rules from data for TS in English. Another task closer to this thesis is question generation (QG) which, to the best of our knowledge, used only the kind of rule-based methods relying on hand-written rules, e.g., (Heilman and Smith, 2010).

In general, a rule-based approach requires some kind of abstraction for formulating rules that could be applied to previously unseen texts. All aforementioned papers relied on abstractions in the terms of *phrase structure* (or constituency) grammars. This abstraction, pioneered by Chomsky (1956), focuses on identifying phrasal constructs (or constituents) in a sentence, e.g., noun phrases or verb phrases, and representing the sentence as a so-called constituency tree. To exemplify, the sentence “You should study these topics or you will fail the exam” could be represented by the constituency tree shown in Figure 3.1. From this figure we can see, for instance, that the two words “the” and “exam” together constitute a noun phrase (NP), and further connected with the verb “fail” create the verb phrase (VP) “fail the exam”. Such structures from constituency trees were used for creating rules for both TS and QG.

There are two major problems with rules based on constituency grammars: (1) they require creating production rules (typically manually), and (2) they are language-dependent. While inducing phrase structure grammars automatically is possible (Brill, 1993), the availability of either such grammars or resources for inducing them for languages beyond English is scarce. The second problem of language-dependent rules is largely inevitable, but the problem becomes worse given that different languages might use different constituents for their grammars.

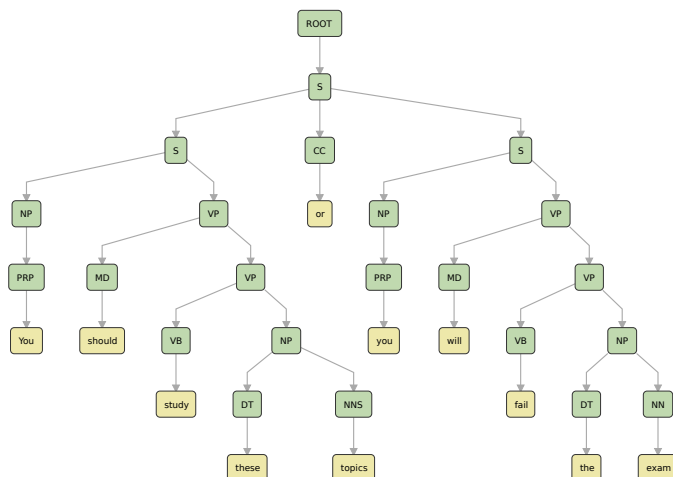


Figure 3.1: The constituency tree for the sentence “You should study these topics or you will fail the exam”, as parsed and visualized by Stanford CoreNLP v4.4.0 (Manning et al., 2014)

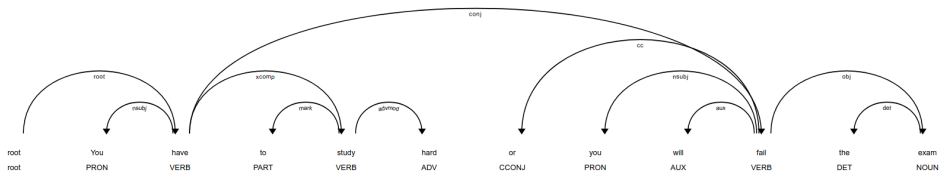


Figure 3.2: The dependency tree for the sentence “You should study these topics or you will fail the exam”, as parsed by Stanza (Qi et al., 2020) and visualized by UDon2 (Kalpakchi and Boye, 2020)

Both aforementioned problems were addressed by the project called *Universal Dependencies* (De Marneffe et al., 2021; Nivre et al., 2016), developed for another abstraction called *dependency grammars*. This formalism describes a sentence in terms of grammatical relations between its words (or lemmas), forming a *dependency tree*. The very same sentence from Figure 3.1 is represented by the dependency tree shown in Figure 3.2. Here we do not see any phrasal constructs, but rather that “the” is a determiner for “exam”, which in its turn is the object of the verb “fail”. The major contribution of Universal Dependencies (UD) was the introduction of the common morphosyntactic annotation scheme for more than 100 languages (as of 2023). This means that while the rules for performing QG, or TS, are likely to remain language-dependent, they can rely on the same abstraction across languages!

3.2 Data-driven approaches

A large portion of research on data-driven approaches in text-to-text NLG, including most of the papers included in this thesis, relies on one particular kind of ML algorithms, namely *artificial neural networks* (ANN). The concept of ANNs dates back to the 1940s (McCulloch and Pitts, 1943) when the researchers took inspiration from the structure of the human nervous system, which is used for operating the brain and exchanging signals between the brain and the rest of the body. One basic component of the nervous system is the electrically excitable cells called *neurons*. On its own, a neuron can do very little. However, when interconnected via synapses to form a *neural circuit* (as they indeed are) they enable humans, among other things, to move, see, talk, and, crucially, learn.

A very simplified description of human learning from the perspective of the nervous system goes like this: each time two biological neurons fire simultaneously while doing a task, their synaptic connection strengthens. This ability to learn is what researchers were trying to simulate when they created *artificial neurons*. Inspired by the biological neurons, the artificial ones can also do very little on their own, for instance, perform simple mathematical operations, like summing all the incoming numbers. However, when interconnected with each other into

artificial neural networks (ANN), they enable computers to achieve much more advanced things, for instance, produce text.

Although research on ANNs started more than 80 years ago, there are still major unanswered questions, such as (1) how many artificial neurons are necessary to solve a given task, (2) how exactly should the neurons be connected (typically referred to as *neural network architecture*, or simply *architecture*), and (3) how should the learning algorithm be simulated (the most widely adopted such algorithm is called *backpropagation* (Rumelhart et al., 1986)). The interested reader is referred to (Goodfellow et al., 2016; Marsland, 2015) for more in-depth information about the field of neural networks.

One way of using neural networks for text-to-text generation is by conforming to the *encoder-decoder framework* (Cho et al., 2014). In this framework, there are two modules: (1) the *encoder* which transforms the incoming text into a fixed-size representation (typically a vector), and (2) the *decoder* that takes this fixed-size representation as input and produces the output text. Both encoder and decoder are often ANNs, and for many years they were of one particular kind, namely *recurrent neural networks*.

Recurrent neural networks

The very basic recurrent neural network can be described by the Equation 3.1, which is visualized in Figure 3.3 for a sequence of two elements. The most appealing aspect of this architecture is its ability to process sequences of theoretically unbounded length by encoding them element-by-element into a fixed-size hidden vector $\mathbf{h}^{(t)}$ for all sequence elements up to and including the t -th one, $\mathbf{x}^{(t)}$.

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}^{(\mathbf{x})}\mathbf{x}^{(t)} + \mathbf{b}^{(\mathbf{x})} + \mathbf{W}^{(\mathbf{h})}\mathbf{h}^{(t-1)} + \mathbf{b}^{(\mathbf{h})}) \quad (3.1)$$

Note that while Equation 3.1 is recurrent, as $\mathbf{h}^{(t)}$ depends on $\mathbf{h}^{(t-1)}$, all associated connections (represented by weights $\mathbf{W}^{(\mathbf{x})}$, $\mathbf{W}^{(\mathbf{h})}$ and biases $\mathbf{b}^{(\mathbf{x})}$, $\mathbf{b}^{(\mathbf{h})}$) are shared across the sequence elements, since they do not depend on t .

Theoretically, RNNs could process sequences of any length. However, in practice, they retain information only from a fixed number of previously processed elements (Bengio et al., 1994). This is due to the gradients becoming either infinitesimally small (vanishing gradients), or very large (exploding gradients), as detailed by Pascanu et al. (2013).

There are many variations of RNNs designed to remedy the aforementioned problems with gradients. The most notable such variations are *long short-term memory networks* Hochreiter and Schmidhuber (1997), and *gated recurrent units* (Cho et al., 2014). Both variants introduce the notion of gates, which are vectors with values between 0 and 1. These gates are used either via element-wise multiplication or via extra addition operations, both of which introduce more paths for the gradient flow, thus reducing the chances of vanishing/exploding gradients.

Beyond the problem with the RNNs, the encoder-decoder framework itself (as we have described it so far) also has a problem. The fixed-size representation

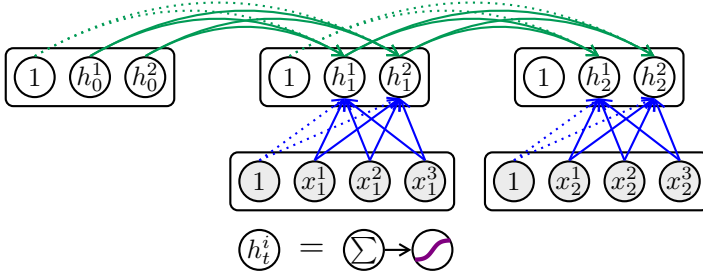


Figure 3.3: The visualization of the recurrent neural network (RNN) from the Equation 3.1. The colors corresponds to the colored variables in the equation, the biases \mathbf{b} are represented as dotted edges. In this example the sequence length is 2, each input is a 3D vector, RNN hidden size is 2.

(FSR) that connects the encoder and the decoder limits the expressiveness of the model and might hinder its performance. As a solution to this problem, Bahdanau et al. (2015) introduced an *attention mechanism*, which introduced one major change. The FSR is now calculated as a weighted sum of the encoder’s hidden states for all input sequence elements. This introduces an extra explicit gradient path to every hidden state of the encoder again helping with avoiding vanishing/exploding gradients.

Transformers

In fact, the attention mechanism turned out to be so powerful that Vaswani et al. (2017) introduced a new architecture called *Transformer*, which is almost entirely based on the attention mechanism. The transformer conforms to the encoder-decoder architecture, where the basic building block for both encoder and decoder is *multi-head attention mechanism* (MHA). Each individual head works as summarized in Figure 3.4. Vaswani et al. (2017) motivate the architecture by viewing any problem as retrieving information represented in the query (represented by the matrix \mathbf{Q} in Figure 3.4) from the set of key-value pairs (represented by matrices \mathbf{K} and \mathbf{V} , respectively in Figure 3.4). Each head i then produces a matrix $\mathbf{A}^{(i)}$ as a result. These matrices are then concatenated for all heads and projected to yet another matrix, which is the final result of the MHA.

Both encoder and decoder in Transformer are built on a specific version of MHA called *multi-head self-attention* (MHSA), which uses the same input matrix as both \mathbf{Q} , \mathbf{K} , and \mathbf{V} . Additionally, the decoder’s MHSA imposes a natural restriction of being able to use only previously decoded tokens, which is called *multi-head masked self-attention*, with the change in computations visualized in Figure 3.5. Finally, there is also multi-head cross-attention between encoder and decoder, calculated exactly as MHA but with $\mathbf{Q} = \mathbf{H}_{dec}$, and $\mathbf{K} = \mathbf{V} = \mathbf{H}_{enc}$.

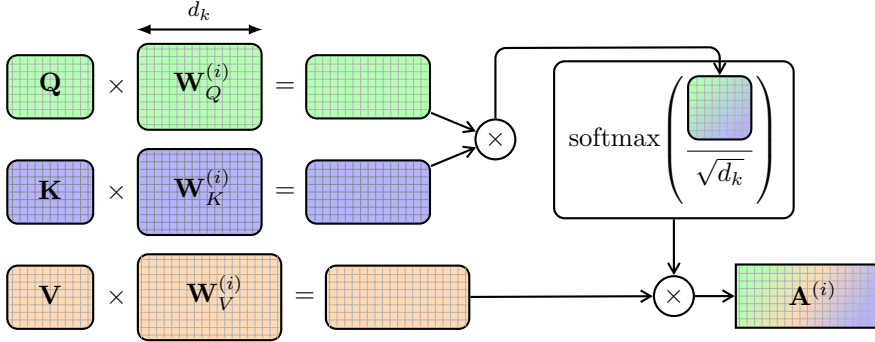


Figure 3.4: The computation diagram for one head of the multi-head attention mechanism. For self-attention, matrices \mathbf{Q} , \mathbf{K} , and \mathbf{V} are represented by one and the same matrix. d_k is the number of columns of $\mathbf{W}_Q^{(i)}$ and $\mathbf{W}_K^{(i)}$

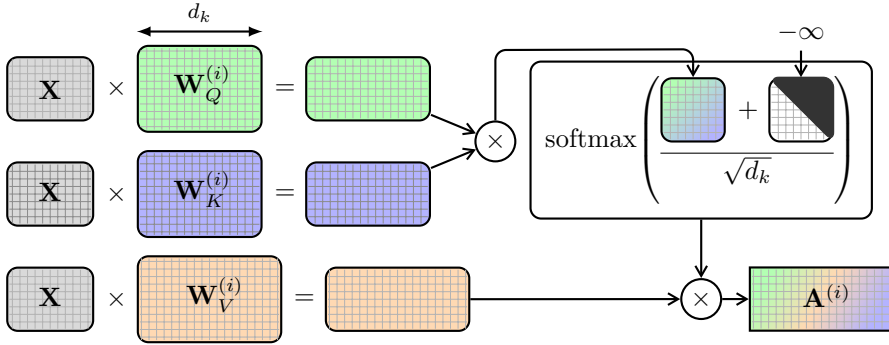


Figure 3.5: The computation diagram for one head of the masked multi-head self-attention mechanism. d_k is the number of columns of $\mathbf{W}_Q^{(i)}$ and $\mathbf{W}_K^{(i)}$

Additional notable architectural differences from RNNs include:

- the introduction of *positional encodings* for keeping track of the order of the tokens in a sequence, which in RNNs was achieved via recurrence;
- many more residual connections ($y = f(x) + x$) providing extra addition operations to improve the gradient flow;
- no recurrence, enabling a higher degree of parallelization.

For further explanations on Transformers we refer to either the original paper (Vaswani et al., 2017), or Jurafsky and Martin (2023, Chapter 10). For more information about the limitations of the original Transformers and ways to remedy them, please refer to the paper by Tay et al. (2022).

Large Language Models (LLMs) were built on the Transformer architecture and were pre-trained on large volumes of unannotated texts, i.e. using self-supervised learning, and then used on the downstream tasks either in a few-shot (zero-shot) manner, or via fine-tuning. There are three LLMs that are of particular relevance to this work: BERT (Devlin et al., 2019), GPT (Radford et al., 2018), and CTRL (Keskar et al., 2019). Training these LLMs first requires training a tokenizer, which splits a text into a number of pieces, called *tokens*, which could be words, parts of words, or even short sequences of letters. The first model, BERT, is based on the Transformer’s encoder and is trained on the following two artificial tasks: (1) restoring corrupted text sequences, and (2) predicting whether two sentences follow each other. The text for BERT is tokenized using the WordPiece tokenizer (Devlin et al., 2019). Texts for the 1st task are corrupted by replacing some tokens with the special [MASK] token. The sentences for the 2nd task are divided using the special [SEP] token. The latter two models, GPT and CTRL, are based on the Transformer’s decoder and are trained on a simpler task of predicting the next token in a text sequence. Both models relied on the tokenizer based on byte-pair encoding (Gage, 1994).

Text representations

So far we have been talking about the structure and kinds of ANNs for text-to-text NLG, but have omitted one crucial aspect, namely how the text should be represented both as input and output for ANNs. Indeed, artificial neural networks operate on numbers, whereas text is a sequence of characters. Although in digital computers, anything is stored as a sequence of numbers (bits), such a representation is not particularly useful for NLG applications. The main reason for this is that it does not encode any information about the meaning of words. For instance, the adjectives **big** and **large** will get absolutely different representations, although they are synonyms.

In order to solve the problem, researchers relied on the *distributional hypothesis*, which states that words appearing in similar contexts have a similar meaning (Firth, 1957). In other words, one can know what a word means by looking at its surroundings. Then if two words (**x** and **y**) tend to co-occur with words that are largely the same, then **x** and **y** must have a similar meaning.

Building upon the distributional hypothesis, one could take a dictionary V , and give each word in V a numerical index. Then one could take a large corpus of texts, and create a word-word matrix \mathbf{W} , where $\mathbf{W}_{i,j}$ gets incremented by 1 if the words with indices i and j co-occur in the context window of size k (k words to the left and k words to the right). To give an example, consider the following very small corpus:

Roses are red, violets are blue, sugar is sweet, and so are you.

For simplicity, let’s consider V to include only words in our corpus, and assign them the following numeric indices:

Roses: 0 are: 2 is: 4 so: 6 sweet: 8 you: 10
 and: 1 blue: 3 red: 5 sugar: 7 violets: 9
 Then the co-occurrence matrix \mathbf{W} for $k = 3$ would look as follows:

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	1	0	1	1	1	0	1
2	0	1	2	1	1	2	1	1	1	2	1
3	0	0	1	0	1	1	0	1	1	1	0
4	0	1	1	1	0	0	1	1	1	0	0
5	0	0	2	1	0	0	0	0	0	1	0
6	0	1	1	0	1	0	0	0	1	0	1
7	0	1	1	1	1	0	0	0	1	1	0
8	0	1	1	1	1	0	1	1	0	0	0
9	0	0	2	1	0	1	0	1	0	0	0
10	0	1	1	0	0	0	1	0	0	0	0

Now a numeric representation of each word from V is a vector corresponding to a specific row in \mathbf{W} . For instance, the word **red** would correspond to $\mathbf{W}_5 = (0, 0, 2, 1, 0, 0, 0, 0, 0, 1, 0)$. When the corpus is large enough, words with similar meaning will get similar vectors (which does not necessarily hold for this small example). However, one problem with such representations is that the matrices become *sparse* (with lots of zeros), which is demonstrated even by this small example. The larger the vocabulary, the sparser the matrix \mathbf{W} will become. Such representation is very space-inefficient, and also very high-dimensional requiring a lot of neurons of ANN, while most of these neurons would receive a value of 0.

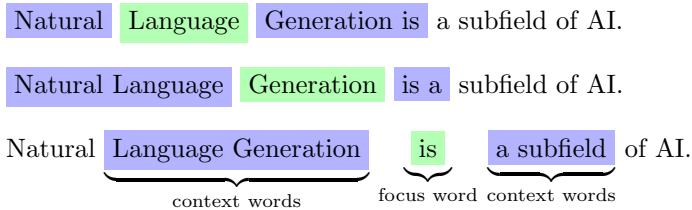
One potential solution to the sparseness problem is to use dimensionality reduction techniques, e.g. Singular Value Decomposition (SVD), Principal Component Analysis (PCA; Pearson (1901)), or t-distributed Stochastic Neighbor Embedding (t-SNE; Van der Maaten and Hinton (2008)). However, while SVD lies at the core of Latent Semantic Analysis (LSA; Dumais et al. (1988)), none of these methods became particularly popular in the neural networks community.

Instead, ANN researchers relied on using *dense* vector representations that are themselves trained using ANNs. Bengio et al. (2000) were the first to propose such an approach, but the idea rose to popularity after Mikolov et al. (2013) proposed their word2vec method. The basic idea behind word2vec is still dividing a text into context windows of size k , but in a slightly different way. Each word can be in two roles: either a focus word (the one currently processed), or a context word (the one within a context window for the focus word). To exemplify, consider a sentence “Natural Language Generation is a subfield of AI”. For the context windows of size $k = 2$, the sentence would be processed as follows.

$\underbrace{\hspace{1.5cm}}$
 focus word

 $\underbrace{\hspace{2.5cm}}$
 context words

Natural
Language Generation is a subfield of AI.



Each word then gets two associated word vectors with it, the ones stored in \mathbf{W} are called focus vectors, and the ones stored in \mathbf{U} are called context vectors. Both these matrices are parameters of ANN trained on the artificial task of predicting the focus word given the context words. Once ANN training is finished, the matrix \mathbf{W} contains word vectors that could be used for training other ANNs for the downstream tasks. After word2vec, researchers have presented a number of approaches improving different aspects of word2vec, for instance, GloVe (Pennington et al., 2014).

One problem with the formulation described above is that only words included in the vocabulary from the very beginning will have a vector. For instance, the word *subfield* might have been in the vocabulary and will have a vector but the word *superfield* will not, since the latter was not present in the training text. However, clearly, both words share a common word stem *field*, but just with different prefixes. This insight was utilized by Mikolov et al. (2018) by learning vectors for parts of the words, which they called *subwords*, instead of only words. The same approach is taken by Transformers, where the subwords, or tokens (e.g. created using the previously discussed tokenizers based on WordPiece or byte-pair encoding) are assigned vectors that are learnt during training. In fact, the input representations (and the output ones, if the representations are tied, e.g., as proposed by Inan et al. (2016)) in Transformers bring one more change: the tokens' position in the text is included in their representation (via the so-called *positional embeddings*). This means that the word *cat* in the phrase *The cat is on the mat* will get a different representation to the *cat* in the phrase *The mat is on the cat*.

3.3 Human evaluation

Evaluating NLG systems is a major challenge, and is not as easy as simply computing accuracy, precision or recall for discriminative tasks. At the time of writing this thesis, the best way of evaluating NLG systems is by setting up a study involving human subjects, which is a complex process that could be roughly divided into the following stages:

1. Decide on the experimental design and pre-register it. This is the most involved stage of the process, for which van der Lee et al. (2021) provides a good summary of the best practices.

2. Write evaluation guidelines that clearly state what should be evaluated, and how.
3. Generate M textual artifacts to be evaluated, and prepare them either in a digital environment or on paper (as per the experimental design).
4. Recruit N human subjects to act as evaluators and execute the experiment.
5. Analyze the results according to the methods specified in the pre-registration.

To the best of our knowledge, there is no consensus on the numbers N and M in the NLG field. There is also no agreement on which criteria should be used, and how the guidelines for their evaluation should be formulated. Neither is there a consensus on how the inter-annotator agreement (IAA) should be estimated. Even in the narrower field of Question Generation (QG), Amidei et al. (2018) report the following wide range of possible values aggregated from the papers between 2013 and 2018.

- The average number of annotators (N) was 4, the most common was 2 (a minimum of 1 and a maximum of 364 annotators)
- The average number of annotated questions (M) was 493 (a minimum of 60 and a maximum of 2186)
- 22 different evaluation criteria (or subsets of these) were used
- 5 different IAA metrics were used (sometimes no IAA analysis at all)

3.4 Automatic evaluation

Evaluating generated text by automatically computing a number of metrics is much simpler, faster, and cheaper than human evaluation. The vast majority of automatic evaluation metrics that are widely used for many text-to-text NLG tasks were initially designed for the task of machine translation. Broadly speaking, these metrics could be divided into the following four groups:

- Group 1. **Based on word overlap**, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), or METEOR (Banerjee and Lavie, 2005).
- Group 2. **Based on edit distance**, such as TER (Snover et al., 2006), CDER (Leusch et al., 2006), or ITER (Panja and Naskar, 2018).
- Group 3. **Partially based on ML models**, where pre-trained or fully-trained ML-models are used to compute *some* parts of the metric, such as MEANT 2.0 (Lo, 2017), YiSi-1 (Lo et al., 2018), BERTScore (Zhang et al., 2019), or as simple as using averages or cosine similarities of word vectors (Sharma et al., 2017).

Group 4. **Fully based on ML models** typically optimized to predict human judgements in the criterion of interest, e.g., BEER (Stanojević and Sima'an, 2014), Blend (Ma et al., 2017), RUSE (Shimanaka et al., 2018), or BLEURT (Sellam et al., 2020).

One major challenge with this approach is that these metrics must remain relevant for showing the quality (or any other desired aspect) of the generated texts for NLG tasks other than machine translation (for which they have most often not been validated).

To illustrate the problem, let us consider the task of question generation (QG) that is relevant to this thesis. Given a text T , we want to generate a question Q that is answerable by T . Let us also consider the following short text:

The Kingdom of Sweden has a population of 10.5 million people.
The capital is Stockholm.

The aforementioned metrics all need a reference set R with instances of correct texts (correct questions, in our case). For our example, let the reference set R consist of the following two questions:

- R1. What is the capital of the Kingdom of Sweden?
- R2. What is the population of Sweden?

Let us also consider that a QG model produced the candidate set C consisting of the following three questions:

- C1. Which city is the Swedish capital?
- C2. How many inhabitants does Sweden have?
- C3. Is the of the?

Let us now discuss how various automatic evaluation metrics behave on this example. We put the most emphasis on the metrics from Group 1, since they are relevant for understanding the papers included in the thesis. Nevertheless, we also briefly discuss metrics from Groups 2, 3, and 4 at the end of this section.

Bilingual evaluation understudy (BLEU)

BLEU represents a family of metrics relying on word overlap between each candidate and the reference set. Single words are called 1-grams (or unigrams), pairs of words 2-grams (or bigrams), triples of words 3-grams (or trigrams), and n -tuples of words n -grams.

BLEU- N denotes a BLEU-metric calculated for up to and including N -grams. Let $c \in C$ be a candidate, and $\hat{r} \in R$ be the best match for c length-wise. Let us also define the following functions: $\text{ng}(c, n)$ returning all n -grams of c , $\text{dng}(c, n)$

returning all *distinct* n -grams of \mathbf{c} , and $\text{len}(\mathbf{c})$ returning the number of words in \mathbf{c} . Then BLEU- N for \mathbf{c} is calculated as follows:

$$\text{BLEU-}N = BP \cdot e^{\sum_{n=1}^N w_n \ln P_n} \quad (3.2)$$

$$BP = \begin{cases} 1, & \text{if } \text{len}(\mathbf{c}) \geq \text{len}(\hat{\mathbf{r}}), \\ e^{1 - \frac{\text{len}(\hat{\mathbf{r}})}{\text{len}(\mathbf{c})}}, & \text{if } \text{len}(\mathbf{c}) < \text{len}(\hat{\mathbf{r}}) \end{cases} \quad (3.3)$$

$$P_n = \frac{\sum_{\mathbf{x} \in \text{dng}(\mathbf{c}, n)} \text{cnt}_{\text{clip}}(\mathbf{x}, \mathbf{c})}{|\text{ng}(\mathbf{c}, n)|} \quad (3.4)$$

$$\text{cnt}_{\text{clip}}(\mathbf{x}, \mathbf{c}) = \min [\text{count}(\mathbf{x}, \mathbf{c}), \max_{\mathbf{r} \in R} [\text{count}(\mathbf{x}, \mathbf{r})]] \quad (3.5)$$

In the equations above BP denotes a *brevity penalty* introduced to penalize candidates that are shorter than the reference translation (since longer candidates have already been penalized by the precision metric).

To exemplify the calculations, let's calculate BLEU-1 to BLEU-4 for the candidate **C1**. First, the clipped counts of up to bigrams calculated using Equation 3.5 are presented below.

$$\begin{aligned} \text{cnt}_{\text{clip}}(\mathbf{Which}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{city}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{is}, \mathbf{C1}) &= \min [1, \max[1, 1]] = 1 \\ \text{cnt}_{\text{clip}}(\mathbf{the}, \mathbf{C1}) &= \min [1, \max[2, 1]] = 1 \\ \text{cnt}_{\text{clip}}(\mathbf{Swedish}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{capital}, \mathbf{C1}) &= \min [1, \max[1, 0]] = 1 \\ \text{cnt}_{\text{clip}}(\mathbf{Which city}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{city is}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{is the}, \mathbf{C1}) &= \min [1, \max[1, 1]] = 1 \\ \text{cnt}_{\text{clip}}(\mathbf{the Swedish}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \\ \text{cnt}_{\text{clip}}(\mathbf{Swedish capital}, \mathbf{C1}) &= \min [1, \max[0, 0]] = 0 \end{aligned}$$

The second step is applying the Equation 3.4 to calculate P_1 , and P_2 .

$$P_1 = \frac{0 + 0 + 1 + 1 + 0 + 1}{6} = 0.5 \quad P_2 = \frac{0 + 0 + 1 + 0 + 0}{5} = 0.2$$

The third step is to calculate the brevity penalty using Equation 3.3. Note that here $\hat{\mathbf{r}}$ is the best-matching reference *by length*, not by content. Hence, $\text{len}(\hat{\mathbf{r}}) = \text{len}(\mathbf{R2}) = 6$. Since the candidate **C1** also has 6 words, there is no brevity penalty, i.e. $BP = 1$.

The final step is calculating BLEU-1, and BLEU-2 using the Equation 3.2. Assuming that the weights $w_n = \frac{1}{N}$, and keeping in mind that $\ln(\cdot)$ denotes a natural logarithm, the calculations proceed as follows.

$$\begin{aligned} BLEU-1 &= 1 \cdot e^{\ln P_1} = P_1 = 0.5 \\ BLEU-2 &= 1 \cdot e^{0.5 \ln P_1 + 0.5 \ln P_2} = e^{0.5 \ln 0.5 + 0.5 \ln 0.2} = 0.3162 \end{aligned}$$

Because there is no trigram from **C1** that is present in R , both BLEU-3 and BLEU-4 are equal to 0.

Performing the same calculations for **C2**, we get the following:

$$P_1 = \frac{0+0+0+0+1+0}{6} = 0.17 \quad P_2 = \frac{0+0+0+0+0}{5} = 0$$

$$\begin{aligned} BLEU-1 &= 1 \cdot e^{\ln P_1} = P_1 = 0.17 \\ BLEU-2 &= 1 \cdot e^{0.5 \ln p_1 + 0.5 \ln p_2} = e^{0.5 \ln 0.17 + 0.5 \ln 0} \rightarrow 0 \end{aligned}$$

Because **C1** and **C2** are equal in length, the brevity penalty also equals to 1. Also similarly, BLEU-3 and BLEU-4 are equal to 0, since **C2** does not share any bigrams with references from R .

Finally, calculating BLEU-1, and BLEU-2 for **C3** looks as follows:

$$P_1 = \frac{1+2+1}{4} = 1 \quad P_2 = \frac{1+0+1}{3} = 0.67 \quad BP = e^{1-\frac{6}{4}} = 0.6065$$

$$\begin{aligned} BLEU-1 &= e^{-0.5} \cdot e^{\ln P_1} = e^{\ln 1 - 0.5} = 0.6065 \\ BLEU-2 &= e^{-0.5} \cdot e^{0.5 \ln P_1 + 0.5 \ln P_2} = e^{0.5 \ln 1 + 0.5 \ln 0.67 - 0.5} = 0.4965 \end{aligned}$$

Similarly to **C1**, the absence of the overlapping trigrams leaves BLEU-3 and BLEU-4 be equal to 0. Before reflecting on what these scores mean (which we do in Section 3.4), let us calculate the other two metrics, ROUGE and METEOR.

Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

Lin (2004) defined four ROUGE-metrics, of which two, ROUGE-N, and ROUGE-S, are based on n -grams, and the other two, ROUGE-L, and ROUGE-W, are based on the longest common subsequence. In this section we will focus on metric that is the most different from BLEU, namely ROUGE-L. Let $LCS(\mathbf{a}, \mathbf{b})$ denote a function for calculating the longest common substring between the strings \mathbf{a} , and \mathbf{b} , then ROUGE-L is calculated as follows.

$$R_{lcs}^{(c,r)} = \frac{\text{len}(LCS(\mathbf{r}, \mathbf{c}))}{\text{len}(\mathbf{r})}; \quad P_{lcs}^{(c,r)} = \frac{\text{len}(LCS(\mathbf{r}, \mathbf{c}))}{\text{len}(\mathbf{c})};$$

$$ROUGE-L(c, r) = \frac{(1 + \beta^2) R_{lcs}^{(c,r)} P_{lcs}^{(c,r)}}{R_{lcs}^{(c,r)} + \beta^2 P_{lcs}^{(c,r)}}$$

$$ROUGE-L(c, R) = \max_{r \in R} ROUGE-L(c, r)$$

The weight β allows to control the relative importance of the precision $P_{lcs}^{(c,r)}$ compared to the recall $R_{lcs}^{(c,r)}$. For this example let us assume that they are equally important ($\beta = 1.0$), and calculate $ROUGE-L(C1, R)$. For this we need to calculate an individual $ROUGE-L$ score for each reference $r \in R$. In turn, this requires calculating the longest common substrings between C1 and each reference r , which is the starting point for the calculations.

$$LCS(C1, R1) = \text{is the} \quad LCS(C1, R2) = \text{is the}$$

$$R_{lcs}^{(C1, R1)} = \frac{\text{len}(\text{is the})}{\text{len}(R1)} = \frac{2}{9}; \quad P_{lcs}^{(C1, R1)} = \frac{\text{len}(\text{is the})}{\text{len}(C1)} = \frac{2}{6};$$

$$R_{lcs}^{(C1, R2)} = \frac{\text{len}(\text{is the})}{\text{len}(R2)} = \frac{2}{6}; \quad P_{lcs}^{(C1, R2)} = \frac{\text{len}(\text{is the})}{\text{len}(C1)} = \frac{2}{6};$$

$$ROUGE-L(C1, R1) = \frac{(1 + 1^2) \cdot \frac{2}{9} \cdot \frac{2}{6}}{\frac{2}{9} + 1^2 \cdot \frac{2}{6}} = 0.2667$$

$$ROUGE-L(C1, R2) = \frac{(1 + 1^2) \cdot \frac{2}{6} \cdot \frac{2}{6}}{\frac{2}{6} + 1^2 \cdot \frac{2}{6}} = 0.3333$$

$$ROUGE-L(C1, R) = \max(0.2667, 0.3333) = 0.3333$$

Since C2 does not have a common subsequence with any of the references in R , $ROUGE-L(C2, R)$ is undefined. On the other hand, C3 overlaps with R , making the following calculations possible.

$$LCS(C3, R1) = \text{is the} \mid \text{of the} \quad LCS(C3, R2) = \text{is the}$$

$$R_{lcs}^{(C3, R1)} = \frac{\text{len}(\text{is the})}{\text{len}(R1)} = \frac{2}{9}; \quad P_{lcs}^{(C3, R1)} = \frac{\text{len}(\text{is the})}{\text{len}(C3)} = \frac{2}{4};$$

$$R_{lcs}^{(C3, R2)} = \frac{\text{len}(\text{is the})}{\text{len}(R2)} = \frac{2}{6}; \quad P_{lcs}^{(C3, R2)} = \frac{\text{len}(\text{is the})}{\text{len}(C1)} = \frac{2}{4};$$

$$ROUGE-L(C3, R1) = \frac{(1 + 1^2) \cdot \frac{2}{9} \cdot \frac{2}{4}}{\frac{2}{9} + 1^2 \cdot \frac{2}{4}} = 0.3077$$

$$ROUGE-L(C3, R2) = \frac{(1 + 1^2) \cdot \frac{2}{6} \cdot \frac{2}{4}}{\frac{2}{6} + 1^2 \cdot \frac{2}{4}} = 0.4$$

$$ROUGE-L(C3, R) = \max(0.3077, 0.4) = 0.4$$

Before reflecting on what these scores mean (which we do in Section 3.4), let us calculate the remaining metric, METEOR.

METEOR

Calculations for METEOR involve four stages, described and exemplified below.

1. Align a candidate and a reference by creating a number of unigram mappings, choosing the one with the most alignments and best preserving the word order. The mapping should be created using exact matches, then stem matches (with Porter stemmer for English), and then synonymy (using WordNet for English)

The aforementioned unigram mappings between each candidate from C and each reference from R are presented in the Figures 3.6, 3.7, 3.8. The aligned unigrams are marked by a vertical bar between them.

Which city is the Swedish capital

What is the capital of the Kingdom of Sweden

Which city is the Swedish capital

What is the population of Sweden

Figure 3.6: The alignments of C1 with R1 (top), and R2 (bottom)

	How	many	inhabitants	does	Sweden	have
What is the	capital	of	the Kingdom	of	Sweden	
	How	many	inhabitants	does	Sweden	have
What is the	population	of	Sweden			

Figure 3.7: The alignments of C2 with R1 (top), and R2 (bottom)

What is the capital of the Kingdom of Sweden

What is the population of Sweden

Figure 3.8: The alignments of C3 with R1 (top), and R2 (bottom)

2. Calculate the F-score as follows:

$$\begin{aligned} m_{c,r} &= \text{count}(\text{aligned unigrams between } c \text{ and } r); \\ n_c &= \text{count}(\text{unigrams of a candidate}); \\ n_r &= \text{count}(\text{unigrams of a reference}); \end{aligned}$$

$$P_{c,r} = \frac{m_{c,r}}{n_c}; \quad R_{c,r} = \frac{m_{c,r}}{n_r}; \quad F_{c,r} = \frac{P_{c,r} \cdot R_{c,r}}{\alpha \cdot P_{c,r} + (1 - \alpha) \cdot R_{c,r}}$$

For the example computations let $\alpha = 0.9$, following Banerjee and Lavie (2005).

$$P_{C1,R1} = \frac{3}{6}; \quad R_{C1,R1} = \frac{3}{9}; \quad F_{C1,R1} = \frac{\frac{3}{6} \cdot \frac{3}{9}}{0.9 \cdot \frac{3}{6} + (1 - 0.9) \cdot \frac{3}{9}} = 0.3448$$

$$P_{C1,R2} = \frac{2}{6}; \quad R_{C1,R2} = \frac{2}{6}; \quad F_{C1,R2} = \frac{\frac{2}{6} \cdot \frac{2}{6}}{0.9 \cdot \frac{2}{6} + (1 - 0.9) \cdot \frac{2}{6}} = 0.3333$$

$$P_{C2,R1} = \frac{1}{6}; \quad R_{C2,R1} = \frac{1}{9}; \quad F_{C2,R1} = \frac{\frac{1}{6} \cdot \frac{1}{9}}{0.9 \cdot \frac{1}{6} + (1 - 0.9) \cdot \frac{1}{9}} = 0.1149$$

$$P_{C2,R2} = \frac{1}{6}; \quad R_{C2,R2} = \frac{1}{6}; \quad F_{C2,R2} = \frac{\frac{1}{6} \cdot \frac{1}{6}}{0.9 \cdot \frac{1}{6} + (1 - 0.9) \cdot \frac{1}{6}} = 0.1667$$

$$P_{C3,R1} = \frac{4}{4}; \quad R_{C3,R1} = \frac{4}{9}; \quad F_{C3,R1} = \frac{\frac{4}{4} \cdot \frac{4}{9}}{0.9 \cdot \frac{4}{4} + (1 - 0.9) \cdot \frac{4}{9}} = 0.4706$$

$$P_{C3,R2} = \frac{3}{4}; \quad R_{C3,R2} = \frac{3}{6}; \quad F_{C3,R2} = \frac{\frac{3}{4} \cdot \frac{3}{6}}{0.9 \cdot \frac{3}{4} + (1 - 0.9) \cdot \frac{3}{6}} = 0.5172$$

3. Calculate the fragmentation penalty ω as follows

$d_{c,r}$ = number of chunks with adjacent matched unigrams in identical order

$$\omega_{c,r} = \gamma \cdot \left(\frac{d_{c,r}}{m_{c,r}} \right)^\beta$$

For this example let $\gamma = 0.5$, and $\beta = 3$, following Banerjee and Lavie (2005).

$$\omega_{C1,R1} = 0.5 \cdot \left(\frac{2}{3} \right)^3 = 0.1481 \quad \omega_{C1,R2} = 0.5 \cdot \left(\frac{1}{2} \right)^3 = 0.0625$$

$$\omega_{C2,R1} = 0.5 \cdot \left(\frac{1}{1} \right)^3 = 0.5 \quad \omega_{C2,R2} = 0.5 \cdot \left(\frac{1}{1} \right)^3 = 0.5$$

$$\omega_{C3,R1} = 0.5 \cdot \left(\frac{2}{4} \right)^3 = 0.0625 \quad \omega_{C3,R2} = 0.5 \cdot \left(\frac{2}{3} \right)^3 = 0.1481$$

4. Calculate METEOR score M as follows:

$$M(\mathbf{c}, \mathbf{r}) = (1 - \omega_{\mathbf{c}, \mathbf{r}}) \cdot F_{\mathbf{c}, \mathbf{r}}; \quad M(\mathbf{c}, R) = \max_{\mathbf{r} \in R} M(\mathbf{c}, \mathbf{r})$$

The final METEOR scores for our example amount to:

$$M(\mathbf{C1}, \mathbf{R1}) = (1 - 0.1481) \cdot 0.3448 = 0.2937$$

$$M(\mathbf{C1}, \mathbf{R2}) = (1 - 0.0625) \cdot 0.3333 = 0.3125$$

$$M(\mathbf{C1}, R) = \max(0.2937, 0.3125) = 0.3125$$

$$M(\mathbf{C2}, \mathbf{R1}) = (1 - 0.5) \cdot 0.1149 = 0.0575$$

$$M(\mathbf{C2}, \mathbf{R2}) = (1 - 0.5) \cdot 0.1667 = 0.0834$$

$$M(\mathbf{C2}, R) = \max(0.0575, 0.0834) = 0.0834$$

$$M(\mathbf{C3}, \mathbf{R1}) = (1 - 0.0625) \cdot 0.4706 = 0.4412$$

$$M(\mathbf{C3}, \mathbf{R2}) = (1 - 0.1481) \cdot 0.5172 = 0.4406$$

$$M(\mathbf{C3}, R) = \max(0.4412, 0.4406) = 0.4412$$

Uses and misuses of automatic evaluation

The scores on automatic metrics against the reference set R for the three candidate questions $\mathbf{C1}$, $\mathbf{C2}$, and $\mathbf{C3}$ are summarized in Table 3.1. For all these metrics, the higher, the better, which makes $\mathbf{C3} \succ \mathbf{C1} \succ \mathbf{C2}$. Recall that supposedly the best match $\mathbf{C3}$ was in fact the question **Is the of the?**, which is hardly a grammatical question to begin with. Both $\mathbf{C1}$ and $\mathbf{C2}$ were both grammatical questions and both were equally on point with respect to the given text. However, in the case of $\mathbf{C2}$, the question **How many inhabitants does Sweden have?** had only one word (**Sweden**) in common with both questions from the reference set R .

This example uncovers one major problem with metrics based on word overlap (Group 1): in order for the comparison with R to be meaningful, R must be an

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
C1	0.5	0.3162	0	0	0.3333	0.3125
C2	0.17	0	0	0	NA	0.0834
C3	0.6065	0.4965	0	0	0.4	0.4412

Table 3.1: Automatic evaluation metrics for the questions from C against the reference set R

exhaustive set presenting all possible instances of correct texts. While designing such a set is mostly possible for machine translation applications (given simple enough sentences), other text-to-text NLG tasks, such as QG, are much more open-ended.

For QG, even defining what *exhaustive* means is not straightforward. To exemplify, recall that thus far we worked with this text:

The Kingdom of Sweden has a population of 10.5 million people.
The capital is Stockholm.

Recall also the questions R1 and C1 that are likely to be included in the same exhaustive set, since they ask about the same thing.

R1. What is the capital of the Kingdom of Sweden?
C1. Which city is the Swedish capital?

Most people would likely agree that both **What is the Sweden’s capital?**, and **What city is the Kingdom of Sweden’s capital?** should be included in that very same exhaustive set. The question **What is the administrative center of Sweden?** is much less clearcut, because this information is not given in the text, but it is true that typically a capital is an administrative center.

Note also that **Sweden** can be substituted by **the Kingdom of Sweden**, **What** can be replaced by either **What city** or **Which city**, **capital** could be converted to **the administrative center** (if there is a consensus about it), and so on. The number of such substitutions is very large, leading to a combinatorial explosion of question reformulations. Additionally every time one makes a substitution, one needs to assess whether the substituted word is a close enough synonym or not, for which there are no clear rules.

To the best of our knowledge, the only QG dataset that attempts to provide an exhaustive list of questions is the Monseratte corpus (Rodrigues et al., 2022). The corpus is in English and is focused only on questions for *single sentences* with, on average, 26 questions associated with each sentence. The list is not guaranteed to be exhaustive and it is indeed very hard to give such a guarantee. In the absence of language resources with exhaustive reference sets, the applicability of automatic evaluation metrics based on word overlap is limited.

The metrics from Group 2 are based on Levenshtein edit distance (Levenshtein et al., 1966), which basically counts the number of simple operations (insertions, deletions, or substitutions) necessary to transform one string into another. The applicability of such metrics for most NLG tasks, QG in particular, is limited by the absence of such exhaustive reference sets, as well.

The metrics from Group 3, and Group 4 would also benefit from exhaustive reference sets. One major difference is that these metrics are based on ML models, which can supposedly represent text in such a way that texts with similar meanings get similar representations. However, evaluating one ML model using

another ML model can lead to a cascading of errors, making it hard to understand whether the reported good (bad) performance is due to errors in the trained model or evaluator model. On top of that, recently, both models can share the same pre-trained Transformer model, e.g., BERT. It is currently unclear whether BERT fine-tuned as the evaluator model could still be used for evaluating the same BERT, but fine-tuned on a different task.

One of the major arguments for using automatic evaluation metrics is that results between different research studies become comparable. However, as we have seen briefly, even the simplest metrics based on word overlap contain a number of hyper-parameters, such as α, β, γ for METEOR, or β for ROUGE-L. Not only could the values of these hyper-parameters differ depending on the task at hand but also the hyper-parameter values are often under-reported in the papers (e.g., Post (2018) reports such a trend for BLEU scores), making the exact comparison impossible in practice.

Chapter 4

State-of-the-art Overview

Recall that the primary objective of this thesis is to automatically generate reading comprehension test items in Swedish. The focus is on conventional multiple-choice test items with four alternatives, of which only one is correct (the key). The problem could be divided into two tasks: (1) generating the stem and the key from the text, later referred to as *Task 1*, and (2) generating three distractors from the text, the stem, and the key, later referred to as *Task 2*.

In this chapter we provide a brief overview of state-of-the-art for the aforementioned problem in 2019 (when we started working on this problem), and compare it to the state-of-the-art in 2023 (at the time of writing of this thesis).

Before 2019

The research advances on generating multiple-choice questions up to and including 2018 are reported in the systematic review by Ch and Saha (2018). To aid the reader, we briefly highlight those of their findings that we deem most relevant for this thesis:

- The generated questions are *often* based only on a single sentence.
- Non-textual information (such as equations, figures, or even bullet points) is mostly *not* taken into account in the generation.
- The authors found no standardized evaluation technique(s) or metric(s), or benchmark datasets for evaluating MCQ generation systems.
- The surveyed papers focused on evaluating stem and key separately from distractors, instead of looking into an MCQ as a whole.
- The majority of the research was conducted for the English language, although few papers dealt also with other languages (notably not Swedish, however).

Regarding the applied methods, to the best of our knowledge, there were no attempts at generating *the whole MCQ at once*, and instead the problem was decomposed into the two tasks, as introduced earlier.

Regarding research for Swedish (which is of core interest to this thesis), to the best of our knowledge, there was no prior research on *data-driven methods* for solving this problem in Swedish, meaning we started with no datasets. The only relevant prior work was on rule-based methods. Indeed, Wilhelmsson (2011, 2012) created the system that could generate question-answer pairs (solving only Task 1). However, a fair comparison to their system is impossible for two reasons. The first one is that their system was evaluated on ten random Wikipedia articles, which unfortunately were not specified. The second one is that the source code of their system is unavailable. This means that we started with no datasets in Swedish and, unfortunately, with no reproducible performance baselines.

As described in Section 3.3, Amidei et al. (2018) reported that there is no agreement in how to conduct human evaluation for Question Generation system. When writing Paper IV we could not find any standardized way of performing human evaluation for Distractor Generation systems either. Instead many researchers relied on Machine Translation metrics, such as those presented in Section 3.4, which is problematic, as we discussed in that section.

Between 2019 and 2023

Generation methodology

The section “Related work” of Papers I and IV provides a review of state-of-the-art up until 2021 for Task 1, and Task 2, respectively. In this section, we briefly review state-of-the-art between 2021 and 2023. The aim of this section is not to provide an exhaustive review but rather to highlight that some issues identified in 2018 remain unresolved in 2023.

Raina and Gales (2022) proposed a method that generates the whole MCQs at once in English. They experimented with a T5 model fine-tuned on RACE++ (Liang et al., 2019), and zero-shot GPT-3 (Brown et al., 2020).

Dijkstra et al. (2022) also proposed a method generating the whole MCQs at once in English also relying on GPT-3, but with fine-tuning it on EQG-RACE (Jia et al., 2021) combined with the original MCQs from RACE (Lai et al., 2017).

Lelkes et al. (2021) presented a method for generating English MCQs based on newspaper articles. They divided the problem into two stages. The first stage was generating stem-key pairs using the fine-tuned PEGASUS (Zhang et al., 2020) model on SQuAD (Rajpurkar et al., 2016), NaturalQuestions (Kwiatkowski et al., 2019), and NewsQA (Trischler et al., 2017). The second stage was generating distractors based solely on the stem *without* including the text or the key by using the fine-tuned T5 (Raffel et al., 2020) to perform “closed-book” question answering on the closed-source dataset.

Rodriguez-Torrealba et al. (2022) focused on generating MCQs from English Wikipedia articles. They proposed to use the pre-trained T5 models in a pipeline manner, first generating stem-key pairs and then distractors. Note that by the nature of their input texts, the vast majority of the generated MCQs will focus *only* on retrieving factual information.

Vachev et al. (2022) also viewed MCQ generation as a two-stage process, and, similarly to Rodriguez-Torrealba et al. (2022), fine-tuned T5 model on SQuAD to first generate stem-key pairs, and then fine-tuned another copy of the T5 model on the dataset of English reading comprehension MCQs called RACE (Lai et al., 2017) to generate distractors. Note that SQuAD is also based on English Wikipedia articles, which means that most of the generated questions will be factual using this approach. Additionally, Zyrianova et al. (2023) showed that a subset of RACE contains MCQs of very different quality levels, including some MCQs with distractors that are not wrong. Hence, RACE alone might not be the best dataset for training (and crucially evaluating) of MCQ generation systems for English.

There are also a number of approaches that attempt to generate only parts of an MCQ. For instance, Shuai et al. (2023) proposed a system for generating the stem and distractors given the text and the key. Their system starts by building a text graph using coreference resolution system. This graph is then encoded and supplied as an input to two custom neural models, one for the stem generation (SG) and one for the distractor generation (DG). Both models are identical and rely on LSTM-based neural networks with hierarchical attention. The models are interconnected during training (by initializing the DG LSTM using the last hidden state of the SG LSTM, and vice versa) in an attempt to capture the relation between the stem and the distractors.

Narayanan et al. (2023) presented a system for generating *only stems* to help caregivers ask open-ended questions on the stories read together with children. They employed a two-step approach by first extracting templates from data based on PoS tags, then applying these templates to unseen sentences and paraphrasing generated (sometimes ungrammatical) questions with pre-trained Transformer model called Parrot (Damodaran, 2021).

Bitew et al. (2022) explored ways of re-using distractors from a pool of MCQs to create new MCQs in English. They proposed to learn a ranking function that assigns a score between 0 (mismatch) and 1 (complete match) to a triple of a stem, the key, and a distractor. The proposed model is based on Multilingual BERT¹

We could not find any research on MCQ generation for Swedish conducted between 2021 and 2023, except our own.

Last, but not least, OpenAI released ChatGPT² in late 2022, which brought substantial performance improvements over previous models, such as GPT-3, for

¹Available at <https://github.com/google-research/bert/blob/master/multilingual.md>

²Announced in the OpenAI’s blog post available at <https://openai.com/blog/chatgpt>

many NLP tasks in various languages. In Paper VII we performed an initial evaluation for reading comprehension MCQ generation in Swedish and (among other things) discuss strengths and weaknesses of ChatGPT for this task.

Evaluation methodology

Some papers (Vachev et al., 2022) relied solely on the Machine Translation (MT) metrics, such as BLEU for evaluating their systems. We have already discussed why this is not such a good idea in Section 3.4. Some papers (Narayanan et al., 2023) relied exclusively on human evaluation, while other papers (Dijkstra et al., 2022; Lelkes et al., 2021; Rodrigues et al., 2022; Shuai et al., 2023) employed a hybrid approach reporting both MT metrics, and conducting human evaluation. It is worth noting that currently there is still no standardized way of conducting such a human evaluation.

Ghanem and Fyshe (2023) proposed a learned metric called DISTO for automatic evaluation of distractors in reading comprehension MCQs. They fine-tuned Distilled RoBERTa (Liu et al., 2019) to encode the following sequence [CLS] [QUES] Q [ANS] A [DIS] D [ART] T, then fed the weight corresponding to the [CLS] token to the sigmoid function. The positive datapoints (with a label of 1) were formed by distractors from the training dataset. The negative datapoints (with a label of 0) were synthesized using various negative sampling approaches. When fine-tuned, DISTO provides a consistency score between stem-key pair (along with the base text) and a supplied distractor, meaning that MCQs with K distractors would need to run DISTO K times. The metric was validated by crowdworkers with the statistically significant Pearson’s correlation of 0.81 between the ratings of crowdworkers and DISTO.

Raina and Gales (2022) also proposed a couple of automatic metrics to assess three different aspects of MCQs, namely unanswerability, diversity, and question complexity. While the first two aspects are assessed using entropy-based metrics, the last one is evaluated using a learned three-way classifier capable of classifying an MCQ as easy, medium or hard. For this purpose the authors proposed to use an ensemble of three ELECTRA (Clark et al., 2020) models trained to predict question complexity based on the partitioning of the RACE++ dataset. As the authors themselves mention, no studies on correlation between the proposed metrics and human judgements have been performed.

Conclusion

As we can see, the MCQ generation field is very active, although mostly for English. Previous research for Swedish is very scarce, which is where this thesis contributes the most. Slowly but surely the field moves from using the MT-based metrics to using human evaluation or designing new automated metrics specifically validated for MCQ generation tasks.

However, the major problem with these new metrics is that they depend on the training data and the employed training hyper-parameters. Not only does this mean that such metrics could be unattainable for less-resourced languages, but also that different training sets and hyper-parameter settings are likely to lead to different scores (and thus a different correlation value with human scores). Another problem is that such metrics are based on models that potentially could be the same models used for the MCQ generation task. In all these instances, it is unclear what such an evaluation of one model with another model would mean. Additionally, the validity of such metrics is often proved by investigating correlations with average human scores, which are not necessarily representative of the obtained human judgments (as inter-annotator agreement tends to be low for MCQ generation tasks). Therefore, in this thesis, we have concentrated on creating faster human evaluations instead (for instance, using single-sentence texts in Paper VII).

Chapter 5

Summary of the Included Papers

All papers included in this thesis share the same broad goal: enabling automatic generation of reading comprehension test items in Swedish. Most of the papers contribute to the narrower goal of generating such test items in the conventional multiple-choice format with four alternatives, which requires generating a stem, the key, and three distractors (introduced in Chapter 2).

Following the terminology introduced in Chapter 3, the pursued problem is a text-to-text NLG task, which could either be solved using a modular or an end-to-end approach. In this thesis we explore both directions, more specifically:

- a) for the modular approach, we split the MCQ generation problem into two subtasks: one for generating the stem and the key (given the text), and another for generating distractors (given the text, the stem, and the key);
- b) for the end-to-end approach, we aim at generating all MCQ components at once (given the text).

The text generation algorithms used throughout the thesis rely on the following two fundamentally different abstractions:

- a) dependency grammars (specifically, Universal Dependencies, briefly introduced in the Section 3.1), which are at the core of the methods presented in Papers I, II, and III;
- b) machine learning models (specifically, Transformer-based neural networks, briefly discussed in the Section 3.2), which constitute the backbone for the methods used in Papers IV, V, VI, and VII.

Note that these two abstractions, while fundamentally different, should not be seen as competing, but rather as completing each other, especially when exploring the modular approach to this problem.

It is worth highlighting that all methods and approaches in the thesis are **data-driven** to a major extent. Even methods based on the dependency grammars, which historically were used for rule-based approaches, are relying on learning the generation templates from data, such as in Papers I, and II.

The crucial component of any data-driven method is, of course, the data, which was non-existent for Swedish when we began exploring this topic. No datasets with multiple-choice questions in Swedish of any kind (let alone, for reading comprehension) have been publicly available. One of the major contributions of this work is, in fact, creating and publishing datasets of multiple-choice *reading comprehension* questions for Swedish. The data was collected using both in a traditional fully manual way using human annotators (such as the SweQUAD-MC dataset from Paper IV), and an alternative way of synthetic data generation with human post-filtering (such as the Quasi dataset synthesized using GPT-3 in the Paper V).

Both of the aforementioned data collection efforts (and many more other unpublished pilot studies) have been fueled by the data collection tool Textinator, developed specifically for this thesis (paper IX). In fact, not only data collection, but also human evaluation was sped up using Textinator for Papers I, II, IV, V, VI, and VII. In Paper IX we provide the description of the tool (which includes the links to the source code, online documentation, and YouTube videos), along with the extensive comparison to other annotation tools for NLP.

Another tool developed for this thesis, is a library for manipulating dependency trees, called UDon2 (Paper VIII). Similarly to Textinator, UDon2 has aided evaluation efforts, albeit for the automatic evaluation, in Paper IV. In fact, both tools fueled all other papers in one way or another, which can be clearly seen in the paper graph in Figure 5.1.

Last, but not least, the evaluation for MCQ generation is still an open challenge. The majority of the included papers rely on human evaluation (designed following the process briefly outlined in Section 3.3).

When designing evaluation criteria for the generated MCQs we have consulted the quality requirements imposed by professional test constructors. For exam-

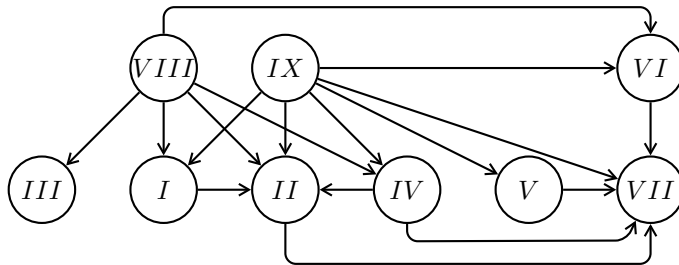


Figure 5.1: Paper graph, where a node X represents a paper numbered X, and the arrow from X to Y means that a paper X has contributed to a paper Y.


ple, (Haladyna, 2004, Table 5.1) lists 31 MCQ-writing guidelines, advising, for instance, to avoid opinion-based or trick MCQs, to keep MCQs independent of each other, to make distractors independent, non-overlapping, of similar length and grammatical structure.


However, when working with NLG models, one has to introduce additional *basic quality requirements*, which the professional test constructors take for granted. These basic requirements include the following:


- the stem, key and all distractors must be grammatically correct;
- the text must be both *sufficient* and *necessary* for finding the key to the stem;
- *all* distractors must indeed be wrong;
- the key must indeed constitute the correct answer for the stem.

These basic requirements, along with those of the text constructors constituted our guidelines for designing human evaluation in the included papers.


The remainder of this chapter provides one-page summaries for each included paper, in which we use the following icons:


 for links to the papers;


 for links to the posters presented at the conferences;

 for links to the associated GitHub repositories;

 for links to the PyPi packages;

 for links to the YouTube channels containing relevant educational videos about the content presented in the paper;

 for links to the websites containing other kinds of resources relevant to the paper (such as online documentation).

The QR codes included in each overview are added for convenience, and point to the papers themselves (similarly to ).

Author contributions

All papers included in this thesis were authored by Dmytro Kalpakchi (D.K.) and Johan Boye (J.B.). For all papers J.B. provided supervision, contributed to writing and proofreading the articles, whereas D.K. was responsible for proposing the methodology, implementation, and writing the manuscripts. Additionally, both authors contributed to the ideation for all papers, and participated in human evaluation for Paper II, whereas human evaluation for Papers V, and VII was performed by D.K. with discussions of challenging cases with J.B.

Paper I overview

Quinductor: A multilingual data-driven method for generating reading-comprehension questions using Universal Dependencies

Dmytro Kalpakchi, Johan Boye

Motivation. Create a *baseline* method for the following problem: given a text T , generate a question-answer pair (q, a) , later referred to as a QA-pair, such that q is the question based on T , and a is the correct answer for q .

Method

1. From the dataset D , select only those QA-pairs, where (1) a_i is a contiguous phrase in the sentence s_i in T_i , and (2) q_i and s_i have at least one word in common.
2. Automatically induce templates that transform s_i into a pair of (q_i, a_i) . The major stage of the induction algorithm relies on a novel variant of shift-reduce algorithm on the linearized dependency tree of s_i . The trees must follow the Universal Dependencies (UD) formalism, introduced in Section 3.1, which is the key to Quinductor’s multilinguality.
3. Given a new text T' , use an overgenerate-and-rank strategy and attempt to apply as many templates from step 4, as possible, and then rank the produced QA-pairs (q', a') using heuristics.

Data. TyDiQA, and SQuAD.

Results

- could induce and apply templates for 5 typologically diverse less-resourced languages;
- performed better than some previously proposed methods for English (especially baselines), while being data-inexpensive and fast to train.

Limitations

- the produced QA-pairs are based on a single sentence;
- the method relies on the consistency of errors in the output of the dependency parser.

 <https://doi.org/10.1017/S1351324923000037>
 <https://github.com/dkalpakchi/quinductor>
 <https://pypi.org/project/quinductor/>



Paper II overview

Automatically generating question-answer pairs for assessing basic reading comprehension in Swedish

Dmytro Kalpakchi, Johan Boye

Motivation. Evaluate how well Quinductor method from Paper I performs for Swedish. Hence, the problem setting is the same as in Paper I.

Method. We have first induced templates for Swedish using Quinductor method on SweQUAD-MC. Then we have judged 9 statements on a Likert scale from 1 (“Disagree”) to 4 (“Agree”), of which the ones of primary interest are:

C1 the question is grammatically correct (↑)

C2 the question makes sense (↑)

C5 the question is relevant to the given sentence (↑)

C6 the suggested answer correctly answers the question (↑)


↑ indicates that the higher the judgements on the Likert scale, the better.


Data. SweQUAD-MC (product of Paper IV).

Results. Among generated QA-pairs:

- 27 of 53 (50.9%) were judged favorably on C1 (median ≥ 3);
- 19 (35.8%) were judged favorably on both C1 and C2 (median ≥ 3);
- 12 (22%) were judged optimally on all criteria.

Limitations. The same as in Paper I.

 <https://arxiv.org/pdf/2211.15568.pdf>

 https://github.com/dkalpakchi/swe_quinductor



Paper III overview

Minor changes make a difference: a case study on the consistency of UD-based dependency parsers

Dmytro Kalpakchi, Johan Boye

Motivation. The basic assumption of the Quinductor algorithm from Paper I is that the dependency parser’s errors are consistent. The aim of this paper is to check how often this assumption holds for state-of-the-art parsers that follow the UD formalism (briefly introduced in Section 3.1).

Method. The experiment is based on sentences that contain 4-digit numerals, such as “John was born in 1992”. For each such *original sentence*, the 4-digit numeral was replaced 50 times by a randomly drawn number between 1100 and 2100, thus creating an *augmented batch* of 50 “new” sentences (e.g., “John was born in 1239”) for each original one.

A consistent dependency parser should then exhibit the following properties:

- if the original is parsed correctly, the augmented batch should also be;
- if the original is parsed incorrectly, all sentences in the augmented batch should have exactly the same kinds of errors.

The study was conducted both on the off-the-shelf Stanza parsers and trained from scratch on the treebanks augmented by either randomly sampling a number of other 4-digit numerals (not used during evaluation), or by replacing all 4-digit numerals in training data with NNNN token.

Data. The Universal Dependencies treebanks from the release v2.8, specifically UD English-EWT, UD Swedish-Talbanken, UD Russian-SynTagRus, UD Ukrainian-IU.

Results

- the errors for off-the-shelf parsers were *inconsistent*;
- the re-trained parsers improved both consistency and performance.

Limitations. The proposed solutions are tested only on 4 languages, all of which are European.

📄 <https://aclanthology.org/2021.udw-1.8.pdf>

🔗 https://github.com/dkalpakchi/ud_parser_consistency



Paper IV overview

BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset

Dmytro Kalpakchi, Johan Boye

Motivation. Assess how well the fine-tuned KB/BERT performs on the following problem: given a text T in Swedish, a stem q based on information present in T , and the key a for q , generate three wrong, but plausible alternative answers (distractors).

Method. Fine-tune KB/BERT on unmasking the distractor tokens either from left to right, or in an arbitrary order (called u-PMLM in the paper). Then generate distractors by unmasking the most likely token, one at a time, in the order the model was trained for.


Data. SweQUAD-MC (collected and released with this paper)


Results


- on average 1.47 out of 3 distractors per MCQ were accepted by the majority among five teachers;
- the most common reason cited by teachers for rejecting a distractor was that it was not wrong (29% of all rejections);
- on average, students correctly selected the key without reading the text for 61% of MCQs with generated distractors (significantly more than the theoretically expected 25%).

Limitations

- The evaluation with students did not check whether the students knew the answer to the question beforehand (without any distractors) making the obtained 61%-level for selecting the key without reading the text harder to interpret.
- Only five teachers have participated in the expert panel evaluating distractors.

 <https://aclanthology.org/2021.inlg-1.43.pdf>

 <https://doi.org/10.5281/zenodo.7966892>

 <https://github.com/dkalpakchi/SweQUAD-MC>



Paper V overview

Quasi: a synthetic Question-Answering dataset in Swedish using GPT-3 and zero-shot learning

Dmytro Kalpakchi, Johan Boye

Motivation. Assess how well GPT-3 performs on the following problem: given the text, generate a specified number of reading comprehension MCQs.

Method. Prompt GPT-3 *in Swedish* without providing any examples, i.e. zero-shot (see the paper for the exact prompt).

Data. 96 texts of varying length, type and genre from the national tests of Swedish for Immigrants courses (swe. SFI nationella prov)

Results. 717 generated MCQs in total, of which 400 were judged to be of insufficient quality, more specifically:

- 31 (4.32%) duplicated;
- 43 (6%) had ungrammatical stems;
- 87 (12.13%) unanswerable (more fine-grained analysis in the paper);
- 20 (2.79%) could be answered without reading the text;
- 10 (1.39%) had at least one ungrammatical alternative;
- 90 (12.55%) had alternatives that made the MCQ irrelevant for the text (more fine-grained analysis in the paper);
- 119 (16.6%) had more than one correct answer.

Note that the problems lower in the list are guaranteed to not have problems higher in the list, but not vice versa. The remaining 317 MCQs on 90 texts constitute Quasi, a synthetic dataset of MCQs released to the research community with additional annotations documented in the paper.

Limitations

- GPT-3 is not open-source

📄 <https://aclanthology.org/2023.nodalida-1.48.pdf>

🔗 <https://github.com/dkalpakchi/Quasi>



Paper VI overview

SweCTRL-Mini: a data-transparent Transformer-based large language model for controllable text generation in Swedish

Dmytro Kalpakchi, Johan Boye

Motivation. Train a model based on the CTRL architecture, capable of controlling the genre of the generated texts in Swedish using control codes.

Method. Pre-train the model from scratch.





Data. Selected parts of Swedish mC4, and texts from Project Runeberg.

Results

- released the fully open source SweCTRL-Mini model (with 494.8 million parameters, and the vocabulary of 256000 tokens) that can be fine-tuned and used for inference on a single consumer-grade GPU;
- released the tool for searching the training data for chunks of text up to 13-grams, and checking if a specific URL was included in the training data;
- performed extensive analysis of the model’s performance, including on SuperLim 2.0.4.

Limitations

- the model produces unformatted text;
- the model produces text with artifacts (remains of HTML, or CSS) despite our best efforts to clean the texts with automatic processing;
- the context window is only 256 tokens, which is not necessarily a problem for simply producing texts, but becomes a problem when fine-tuning the model for some tasks where the whole text is required, for instance, text summarization.

 <https://arxiv.org/abs/2304.13994>
 <https://github.com/dkalpakchi/SweCTRL-Mini>
 <https://swectrl.dev/>
 <https://huggingface.co/dkalpakchi/SweCTRL-Mini>



Paper VII overview

Generation and Evaluation of Multiple-choice Reading Comprehension Questions for Swedish

Dmytro Kalpakchi, Johan Boye

Motivation. Compare the performance of the fine-tuned versions of openly available models for Swedish (KB/BERT and SweCTRL-Mini) with GPT-3 and ChatGPT (along with some baselines) on the problem from Paper V.

Method. Fine-tuned KB/BERT and SweCTRL-Mini on two different datasets. The fine-tuned models were compared to two baselines: UD-based one, and zero-shot SweCTRL-Mini. All proposed models (ten in total) were then compared to the publicly available state-of-the-art models (at the time of writing), namely GPT-3 and ChatGPT.

Data. SweQUAD-MC, Quasi, specifically designed single-sentence texts (SSTs), Plugga (the latter two released with this paper).

Results

- ChatGPT performs substantially better than *all* other models in *all* evaluation settings. The runner up is GPT-3, and SweCTRL-Mini trained on SweQUAD-MC is third.
- The human evaluation results based on the toy domain of SSTs closely resemble those on Plugga, indicating that SST-based evaluation might be a viable lower-cost alternative to the full-scale human evaluation (but more extensive studies on the matter are needed).

Limitations

- Hyper-parameter tuning for non-GPT models was manual and very limited, which means that their performance could be boosted further (although unlikely to reach ChatGPT).
- The evaluation was mainly focused on uncovering foundational problems with the generated MCQs, meaning that even MCQs judged to be acceptable might not necessarily be of sufficient quality to be useful in a high-stakes testing scenario.



📄 <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-329400>

🔗 <https://github.com/dkalpakchi/MCQ-Gen>

Paper VIII overview

UDon2: a library for manipulating Universal Dependencies trees

Dmytro Kalpakchi, Johan Boye

UDon2 is an open-source library for manipulating dependency trees compatible with the Universal Dependencies. UDon2 is aimed at developers of downstream Natural Language Processing applications that require manipulating dependency trees on the sentence level (to complement other available tools geared towards working with treebanks).

 <https://aclanthology.org/2020.udw-1.14.pdf>
 <https://doi.org/10.5281/zenodo.7966929>
 <https://github.com/udon2/udon2>
 <https://pypi.org/project/udon2/>
 <https://udon2.github.io/>








Paper IX overview

Textinator: an Internationalized Tool for Annotation and Human Evaluation in Natural Language Processing and Generation

Dmytro Kalpakchi, Johan Boye

Textinator allows annotating data for a wide variety of NLP tasks, and its user interface is offered in multiple languages, lowering the entry threshold for domain experts. The paper presents a thorough systematic comparison of Textinator to previously published annotation tools along 9 different axes.

 <https://aclanthology.org/2022.lrec-1.90.pdf>
 <https://doi.org/10.5281/zenodo.7966905>
 <https://github.com/dkalpakchi/Textinator>
 <https://www.youtube.com/@textinator/>
 <https://textinator.readthedocs.io/en/latest/>



Chapter 6

Discussion and Conclusions

6.1 Limitations

There are three major limitations to understand about this work. The first one is that we investigated how well automatic methods could be used for designing a good MCQ *on a single text*. This excludes, for instance, MCQs requesting to compare different text, or find the most appropriate text for the title. Furthermore, the designed MCQ generation methods use only textual information **ignoring formatting, or any visuals**, such as images, graphs, or charts.

The second thing to realize is that the focus of this work is on designing a good MCQ, and not a good test with a number of MCQs. Combining MCQs into a meaningful test typically requires determining some order that is not random. Whether such ordering has an effect in practice and can be optimized is out of the scope of this thesis.

The third and final thing to realize is that an essential component of any reading comprehension MCQ is the text itself, which needs to be on the appropriate level. In this work, we did not analyze or generate the texts themselves and assumed that the given texts are on the appropriate level. The main reason for this is that the appropriate level is dictated by factors out of our control, namely reader factors (background, education, professional experience, knowledge of other languages, etc.) and task factors (time constraints, high-stakes or low-stakes, digital or pen-and-paper, etc), which are further discussed by OECD (2019). Furthermore, the Swedish for Immigrants courses (SFI) accept students of all kinds on a rolling basis, making it impossible to fix the reader characteristics that would describe an average student. Instead, we focused on checking whether the generated MCQs could work with the given text and crucially did not inquire about anything beyond the information in the text. Beyond these three central limitations, there are also minor ones worth highlighting.

Regarding the evaluation of the explored generation methods, in most cases, we did not conduct evaluations with teachers. The only exception was the distractor generation method proposed in Paper IV. This evaluation made us realize that these methods had much more foundational problems that needed to be fixed before we could run another evaluation with teachers. Between the start of my Ph.D. studies in 2018, and their finish in 2023, state-of-the-art NLG methods have advanced substantially, which is why we attempted generating the whole MCQs at once. However, as we discovered in Papers V, and VII, many foundational problems should be addressed before asking teachers to invest their time and expertise in evaluating the MCQs.

Regarding texts themselves, we briefly touched upon how well MCQ generation systems stick to the texts, both in cases when it is desirable (e.g., using the spelling from the text if alternative spellings are possible), and undesirable (e.g., when the text contains OCR errors). Analyzing and accounting for other types of mistakes, such as factual errors, is beyond the scope of the thesis. The rationale is that these other types of mistakes impact the quality of the text to a degree that we might not even need consider generating MCQs on such texts. Instead such texts should be detected and discarded or corrected, which is also beyond the scope of this thesis.

6.2 Why not translate automatically?

As we have seen in Chapter 4, English has much more language resources that could be used for MCQ generation (including both datasets and pre-trained models). A natural question would be why not simply automatically translate Swedish texts to English, generate MCQs in English, and then automatically translate them back to Swedish?

There are multiple problems with this approach. The first one is that translations are rarely exact, some words are simply non-translatable in a one-to-one match (e.g., *lagom* in Swedish), and sometimes subtle aspects of the meaning of words and phrases might get lost changing the perception of text. The second problem is that phrases and constructs from the source language could leak into the target language. Gellerstam (1986) studied such effects on the English translations of Swedish novels and referred to this effect as Translationese. The interested reader is referred to the empirical study by Koppel and Ordan (2011) for more discussion on the matter.

6.3 Societal impact

The research in this thesis is about helping computers to help assess humans. As any kind of investigation into automation techniques, it raises concerns about automating away people, thus increasing unemployment and negatively impacting people’s lives and livelihoods. To address this concern, I want to stress that the

goal of this research is only to *help* assess, very much preserving a human (in this case a teacher) in the loop. This is because the role of a teacher goes beyond conveying information and testing the student's knowledge. The central role of a teacher is to spark interest in learning something and encourage the pursuit of this interest further. This is something that is neither possible nor indeed desirable to automate.

Another concern is that having access to an easy-to-use and fast system for generating MCQs would encourage and give tools for a “learning to test”-mindset. In many cases, the tests, especially in multiple-choice format, are involved in high-stakes testing, not least at SFI courses, where passing or failing the course could lead to a residence permit being granted or rejected. As discussed by Jones (2007), the widespread use of high-stakes testing has had some unintended outcomes, such as comparing educational establishments based on the test results (which is not entirely fair given that students from different backgrounds develop at different paces), or shifting the whole goal of education to just passing standardized tests. One way to remedy this (which I personally believe in), is to use such tests not only for summative, but also for formative assessment. For instance, a test could be an entry point for discussing the text, or it could reveal which parts of the text were misunderstood followed by discussing why. In this setting, the fast MCQ generation system would be desirable to let teachers use the texts they find appropriate, instead of relying on a fixed set of didactic materials, or old national exams.

6.4 Contributions towards Open science

In all papers included in this thesis, we have attempted to follow the Open science movement, which has manifested itself in the following ways:

- We used publicly available datasets, and publish our own datasets free of charge (two datasets, SweQUAD-MC¹, and Quasi², collected throughout the thesis are publicly available on GitHub with mirrors on Zenodo).
- We were explicit about research methodology, e.g., by pre-registering³ human evaluations involving hypothesis testing, or by reporting the exact human evaluation guidelines in the published papers.
- We made all research code open source, including the developed platform for data collection, Textinator⁴.

¹<https://github.com/dkalpakchi/SweQUAD-MC>

²<https://github.com/dkalpakchi/Quasi>

³Only Paper IV involved doing such a study with a pre-registration available at <https://osf.io/b7pjm>

⁴<https://github.com/dkalpakchi/Textinator>

- We made the research results accessible, for instance, by publishing video tutorials⁵ or extensive documentation⁶ for the developed tools, or making the trained foundational model, SweCTRL-Mini, accessible online⁷.
- We made the training data for SweCTRL-Mini easily searchable⁸ to the extent that one could check whether the text from a specific URL was included in the training data.

6.5 Contributions towards Sustainability

This thesis contributes directly to the goals 4 (Quality Education) and 10 (Reduced Inequalities) of the United Nations' Sustainable Development Goals (SDG). Developing reliable MCQ generation system for reading comprehension helps improving the examination process, and also could be used for formative assessment to help students learn. Ultimately both of these goals will help teachers understand the needs of their students better, and thus lead to higher quality education in line with the SDG 4. At the same time, being focused on Swedish for Immigrants courses, the research in this thesis is a contribution towards a better integration of immigrants into Swedish society, thus reducing the inequality in line with the SDG 10.

Additionally, this thesis contributes indirectly towards SDG 12 (Responsible Consumption and Production) by focusing on lightweight methods to minimize the carbon footprint from using them. This manifested itself in three ways:

- attempting to fine-tune or use out-of-the-box previously trained models first;
- developing methods other than energy-hungry neural networks;
- if the above approaches were infeasible, training neural networks that could be used for fine-tuning and inference on a single consumer-grade GPU.

6.6 Future work

This section is intentionally left blank.

⁵<https://www.youtube.com/@textinator>

⁶<https://udon2.github.io/>

⁷<https://huggingface.co/dkalpakchi/SweCTRL-Mini>

⁸<https://swectrl.dev/data>

6.7 Conclusions

When we started this journey, there were neither datasets, nor methods for doing MCQ generation in Swedish. Now, at the end of this journey, there are:

- two datasets for MCQ generation, SweQUAD-MC, and Quasi;
- one tool for collecting NLP data, Textinator, which is open-source and deployable using Docker;
- a number of openly available methods for generating MCQs as a whole or only separate parts (introduced in Papers I, IV, and VII);
- a methodology for evaluating generated MCQs (introduced in Papers I, IV, V, and VII).

The papers included in this thesis were produced during three and a half years of work. This is because the topic of the thesis changed radically after the first year, meaning that the article produced during that year became completely irrelevant for the present version of the thesis. Nevertheless, despite being a bit short on time, I believe that the work presented in this thesis has laid solid foundations for further research on designing and, crucially, evaluating MCQ generation methods for Swedish.

References

- Amidei, J., Piwek, P., and Willis, A. (2018). Evaluation methodologies in automatic question generation 2013-2018.
- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Banaee, H., Ahmed, M. U., and Loutfi, A. (2013). Towards nlg for physiological data monitoring with body area networks. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 193–197.
- Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.
- Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, 13.
- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.
- Binkley, M. R. and Rust, K. (1994). *Reading Literacy in the United States: Technical Report of the US Component of IEA Reading Literacy Study*. National Center for Education Statistics, US Department of Education, Office of Educational Research and Development.
- Bitew, S. K., Hadifar, A., Sterckx, L., Deleu, J., Develder, C., and Demeester, T. (2022). Learning to reuse distractors to support multiple choice question generation in education. *IEEE Transactions on Learning Technologies*.
- Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *31st Annual Meeting of the Association for*

- Computational Linguistics*, pages 259–265, Columbus, Ohio, USA. Association for Computational Linguistics.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Nee-lakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Ch, D. R. and Saha, S. K. (2018). Automatic multiple choice question generation from text: A survey. *IEEE Transactions on Learning Technologies*, 13(1):14–25.
- Chandrasekar, R. and Srinivas, B. (1997). Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.
- Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.
- Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Council of Europe (2001). *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press.
- Damodaran, P. (2021). Parrot: Paraphrase generation for nlu.
- De Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal dependencies. *Computational linguistics*, 47(2):255–308.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dijkstra, R., Genç, Z., Kayal, S., Kamps, J., et al. (2022). Reading comprehension quiz generation using generative pre-trained transformers.
- Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., and Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285.

- El-Kassas, W. S., Salama, C. R., Rafea, A. A., and Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679.
- Firth, J. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, pages 10–32.
- Gage, P. (1994). A new algorithm for data compression. *C Users J.*, 12(2):23–38.
- Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.
- Gellerstam, M. (1986). Translationese in swedish novels translated from english. *Translation studies in Scandinavia*, 1:88–95.
- Ghanem, B. and Fyshe, A. (2023). Disto: Evaluating textual distractors for multi-choice questions using negative sampling based approach. *arXiv preprint arXiv:2304.04881*.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Haladyna, T. M. (2004). *Developing and validating multiple-choice test items*. Routledge.
- Haladyna, T. M. and Downing, S. M. (1993). How many options is enough for a multiple-choice test item? *Educational and psychological measurement*, 53(4):999–1010.
- Haladyna, T. M., Downing, S. M., and Rodriguez, M. C. (2002). A review of multiple-choice item-writing guidelines for classroom assessment. *Applied measurement in education*, 15(3):309–333.
- Heilman, M. and Smith, N. A. (2010). Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Inan, H., Khosravi, K., and Socher, R. (2016). Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.
- Jia, X., Zhou, W., Sun, X., and Wu, Y. (2021). Eqg-race: Examination-type question generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):13143–13151.

- Jones, B. D. (2007). The unintended outcomes of high-stakes testing. *Journal of applied school psychology*, 23(2):65–86.
- Jurafsky, D. and Martin, J. H. (2023). *Speech and Language Processing (3rd Edition, draft)*. Internet: <https://web.stanford.edu/~jurafsky/slp3/>, [Accessed: 2023-05-15].
- Kalpakchi, D. and Boye, J. (2020). UDon2: a library for manipulating Universal Dependencies trees. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 120–125, Barcelona, Spain (Online). Association for Computational Linguistics.
- Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.
- Koppel, M. and Ordan, N. (2011). Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1326, Portland, Oregon, USA. Association for Computational Linguistics.
- Krahmer, E. and Van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.
- Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017). RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.
- Lelkes, A. D., Tran, V. Q., and Yu, C. (2021). Quiz-style question generation for news stories. In *Proceedings of the Web Conference 2021*, pages 2501–2511.
- Leppänen, L., Munezero, M., Granroth-Wilding, M., and Toivonen, H. (2017). Data-driven news generation for automated journalism. In *Proceedings of the 10th international conference on natural language generation*, pages 188–197.
- Leusch, G., Ueffing, N., and Ney, H. (2006). CDER: Efficient MT evaluation using block movements. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 241–248, Trento, Italy. Association for Computational Linguistics.

- Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.
- Liang, Y., Li, J., and Yin, J. (2019). A new multi-choice reading comprehension dataset for curriculum learning. In Lee, W. S. and Suzuki, T., editors, *Proceedings of The Eleventh Asian Conference on Machine Learning*, volume 101 of *Proceedings of Machine Learning Research*, pages 742–757. PMLR.
- Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Lo, C.-k. (2017). MEANT 2.0: Accurate semantic MT evaluation for any output language. In *Proceedings of the Second Conference on Machine Translation*, pages 589–597, Copenhagen, Denmark. Association for Computational Linguistics.
- Lo, C.-k., Simard, M., Stewart, D., Larkin, S., Goutte, C., and Littell, P. (2018). Accurate semantic textual similarity for cleaning noisy parallel corpora using semantic machine translation evaluation metric: The NRC supervised submissions to the parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 908–916, Belgium, Brussels. Association for Computational Linguistics.
- Ma, Q., Graham, Y., Wang, S., and Liu, Q. (2017). Blend: a novel combined MT metric based on direct assessment — CASICT-DCU submission to WMT17 metrics task. In *Proceedings of the Second Conference on Machine Translation*, pages 598–603, Copenhagen, Denmark. Association for Computational Linguistics.
- Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Marsland, S. (2015). *Machine learning: an algorithmic perspective*. CRC press.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.
- Mikolov, T., Grave, E., Bojanowski, P., Puhersch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.
- Narayanan, A. B. L., Gomez, L. E., Fernandez, M. M. S., Nguyen, T., Blais, C., Restrepo, M. A., and Glenberg, A. (2023). Genq: Automated question generation to support caregivers while reading stories with children. *arXiv preprint arXiv:2305.16809*.
- Nivre, J., De Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.
- OECD (2019). *PISA 2018 Assessment and Analytical Framework*. OECD Publishing, Paris.
- OECD (2021). *PISA 2025 Foreign Language Assessment Framework*. OECD Publishing, Paris.
- Panja, J. and Naskar, S. K. (2018). ITER: Improving translation edit rate through optimizable edit costs. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 746–750, Belgium, Brussels. Association for Computational Linguistics.
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.
- Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.
- Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

- Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.
- Raina, V. and Gales, M. (2022). Multiple-choice question generation: Towards an automated assessment framework. *arXiv preprint arXiv:2209.11830*.
- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Rodrigues, H., Nyberg, E., and Coheur, L. (2022). Towards the benchmarking of question generation: introducing the Monserrate corpus. *Language Resources and Evaluation*, 56(2):573–591.
- Rodriguez-Torrealba, R., Garcia-Lopez, E., and Garcia-Cabot, A. (2022). End-to-end generation of multiple-choice questions using text-to-text transfer transformer models. *Expert Systems with Applications*, 208:118258.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Saha, S., Zhang, S., Hase, P., and Bansal, M. (2022). Summarization programs: Interpretable abstractive summarization with neural modular trees. *arXiv preprint arXiv:2209.10492*.
- Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.
- Sharma, S., Asri, L. E., Schulz, H., and Zumer, J. (2017). Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.
- Shi, W., Zhou, H., Miao, N., and Li, L. (2020). Dispersed exponential family mixture vaes for interpretable text generation. In *International Conference on Machine Learning*, pages 8840–8851. PMLR.

- Shimanaka, H., Kajiwar, T., and Komachi, M. (2018). RUSE: Regressor using sentence embeddings for automatic machine translation evaluation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 751–758, Belgium, Brussels. Association for Computational Linguistics.
- Shuai, P., Li, L., Liu, S., and Shen, J. (2023). Qdg: A unified model for automatic question-distractor pairs generation. *Applied Intelligence*, 53(7):8275–8285.
- Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.
- Stanojević, M. and Sima'an, K. (2014). BEER: BETter evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.
- Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G., and Cucchiara, R. (2022). From show to tell: A survey on deep learning-based image captioning. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):539–559.
- Suter, J., Ebling, S., and Volk, M. (2016). Rule-based automatic text simplification for german.
- Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2022). Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.
- Trischler, A., Wang, T., Yuan, X., Harris, J., Sordani, A., Bachman, P., and Suleman, K. (2017). NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.
- Vachev, K., Hardalov, M., Karadzhov, G., Georgiev, G., Koychev, I., and Nakov, P. (2022). Leaf: Multiple-choice question generation. In *European Conference on Information Retrieval*, pages 321–328. Springer.
- van der Lee, C., Gatt, A., van Miltenburg, E., and Krahmer, E. (2021). Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, 67:101151.
- Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

- Wilhelmsson, K. (2011). Automatic question generation from swedish documents as a tool for information extraction. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 323–326.
- Wilhelmsson, K. (2012). Automatic question generation for swedish: The current state. In *Proceedings of the SLTC 2012 workshop on NLP for CALL; Lund; 25th October; 2012*, number 080, pages 71–79. Linköping University Electronic Press.
- Wiseman, S., Shieber, S., and Rush, A. (2018). Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.
- Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). BERTScore: Evaluating text generation with BERT. *arXiv preprint arXiv:1904.09675*.
- Zhou, J. and Bhat, S. (2021). Paraphrase generation: A survey of the state of the art. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5075–5086.
- Zyrianova, M., Kalpakchi, D., and Boye, J. (2023). Embrace: Evaluation and modifications for boosting race. *arXiv preprint arXiv:2305.08433*.