MASTER'S THESIS 2023

# Correction of Grammatical Errors in Swedish

Joel Ehnroth, Yoonjoo Park


ISSN 1650-2884
LU-CS-EX: 2023-29

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-29

# Correction of Grammatical Errors in Swedish

## Korrigering av grammatiska fel på svenska

**Joel Ehnroth, Yoonjoo Park**

# Correction of Grammatical Errors in Swedish

Joel Ehnroth
jo3268eh-s@student.lu.se

Yoonjoo Park
yo4302pa-s@student.lu.se

June 27, 2023

Master's thesis work carried out at NordAxon AB.

Supervisors: Filip Bolling, filip.bolling@nordaxon.com
Pierre Nugues, pierre.nugues@cs.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

**Abstract**

This thesis investigates Swedish grammatical error detection and grammatical error correction using transformer-based models. The models are evaluated on a human-annotated parallel dataset containing incorrect-correct sentence pairs. We explore several pre-trained large language models, both monolingual and multilingual variants, that we fine-tune on the task-specific dataset DaLAJ-GED. We evaluate the models' ability to classify sentences as correct or incorrect, identify where in the sentence an error occurs and classify what type of error it is. Furthermore, we evaluate different models' ability to generate a corrected version of a grammatically incorrect sentence. To this end, we propose combining the limited human-annotated data with synthetic data to improve model performance on these tasks. Our results show that transformer-based models outperform traditional rule-based methods on error detection and correction. Specifically, when evaluating on the DaLAJ-GED dataset for the correction task, the Swedish transformer model GPT-SW3 achieved 80% accuracy compared to the 52% accuracy of a leading Swedish rule-based model. Our work contributes to the field by showing the potential of transformer-based models, and proposing several ways for future development of these systems.

**Keywords**: natural language processing, transformers, grammatical error detection, grammatical error correction

# Acknowledgements

We would like to express gratitude to our supervisor at NordAxon, Filip Bolling, whose valuable input has guided our work over the course of the project. We are also grateful to our supervisor at Lund University, Professor Pierre Nugues, for having provided us with his academic experience which has ensured a steady progression of our work, as well as for his continuous encouragement. We also wish to thank our examiner, Jacek Malec.

Finally, we would like to thank everyone at the NordAxon office for giving us the opportunity to explore this subject matter, and for re-affirming the value of our work.

# Contents

# Chapter 1
# Introduction

## 1.1 Motivation and Goals

Written communication remains a key aspect of life in the digital age, and the ability to express oneself fluently and precisely is important, whether it is for professional letters and reports, job applications or simple daily communication through emails or text messages. Tools that provide grammatical feedback can play an instrumental role in helping people achieve their goals, and this may be particularly important for second-language learners. Unfortunately, the field of grammatical error detection (GED) and grammatical error correction (GEC) is dominated by the major languages such as English, but is comparatively under-explored for the Swedish language.

The complexity of human languages makes the task of providing precise GEC difficult. Our natural languages typically involve a large vocabulary of words whose meanings are context-dependent, meaning that the semantics of a word can vary depending on what other words surround it. There are also many different forms of sentence building blocks, called parts of speech. These include nouns, verbs, adjectives, prepositions, pronouns and more, and they all interbehave in different ways. Grammatical errors involving nouns can look different from grammatical errors involving verbs. Even within each part of speech group, there exists a great variation.

Here is an example of an error.

**Sentence:** Vi flyttade i Sverige och vi bor ett hus nu.
**Correction:** Vi flyttade till Sverige och vi bor i ett hus nu.

In the original sentence, the preposition *i* after *flyttade* is corrected to *till*. On the other hand, after the verb *bor*, the preposition *i* is added. It shows that the use of a preposition depends on the verb.

Here is another example.

**Sentence:** Vi bor i hotellet.
**Correction:** Vi bor på hotellet.

In the previous example, the verb *bor* was followed by the preposition *i*, but in this example *i* requires the correction *på*, because the error now involves the noun *hotellet*. All of this contributes to GEC being a difficult task to solve.

Linguists have long debated whether language can be fully described by consistent rules, and in any case, the implementation of a sufficiently fine-tuned and complex ruleset has proven difficult. For this reason, recent research in the field has instead turned to big data, which is fed to machine learning models called neural networks. The hope is that, by showing the models vast quantities of text data, they can form an internal model of the language(s) seen in the data. Indeed this approach has proven to be able to generate convincing text, as models like ChatGPT have shown.

Transformers are a type of neural network that have achieved state-of-the-art results on most tasks involving natural language processing (NLP). They are used today in a wide range of consumer-facing products, from grammar correction services to chatbots and text summarizers to virtual assistants and more. Their distinguishing feature is their ability to process sequences of data and paying attention to the entirety of the sequence whilst processing each part of it. This is particularly important in NLP, as the semantic meaning of a word can be determined by words that precede or follow it, hence it is important that the network can pay attention to the words that surround a word, as it processes it. In short, the transformer learns the importance of each word in a sentence, for every other word in the same sentence.

In this thesis, we investigate recent attempts to employ neural networks, specifically transformer models, for the tasks of GED and GEC on Swedish texts. The former involves detecting if a sentence contains an error, where in the sentence the error occurs, and what type of error it is. The latter involves generating a corrected version of the sentence. We explore current state-of-the-art methods and attempt to improve on previous approaches for GEC of the Swedish language.

The work was carried out in collaboration with NordAxon, a company working to leverage the latest advancements in machine learning to give people in Sweden tools to help them succeed in communicating effectively. Their product, Emely, is an AI-driven conversational assistant designed in part for second-language learners of Swedish, to offer a safe environment in which users can practice communicating in the Swedish language at their own pace. A key part of such a system could potentially be a module that reviews spelling, grammar, punctuation and more, in the user's input texts, and proposes changes for the identified problems.

## 1.2 Research Questions

This thesis aims to answer the following questions:

- What tools and resources exist for GED and GEC in Swedish, and how do they compare to each other?

- Is there any merit to combining old and new techniques of NLP to improve GED and GEC performance with a focus on precision?
- How do different types of transformer architectures perform on the task of GED and GEC?
- Which types of grammatical errors are difficult to detect and correct? How can we resolve these issues?

## 1.3 Scope

This scope of the thesis will be delimited by the following:

- We will only test a selected number of transformer models.
- We will not consider inference times or the feasibility of deployment in production.
- We will only investigate GED and GEC for the Swedish language.

## 1.4 Contributions

We contribute to the field of GED and GEC for the Swedish language by evaluating modern transformer-based models trained on the latest datasets available for the Swedish language. We incorporate recent research on synthetic data and some different approaches to GEC based both on encoder-decoder models and pure decoder-models. Both authors have contributed to every part of this thesis, working in tandem on every experiment and every part of the report.

## 1.5 Outline

The thesis is structured as follows. In Chapter 2, we describe the datasets used for our experiments.

In Chapter 3, we describe relevant research that has previously been done in the field of GED and GEC, as well as work that has been done on generating synthetic data.

In Chapter 4, we present the theoretical background on transformer models needed to understand the work that has been carried out.

In Chapter 5, we describe the setup and method for the experiments with GED and GEC that have been carried out. We further describe our procedure for generating synthetic data that has been used in the experiments.

In Chapter 6, we present the results from our experiments and compare the performance of traditional methods with contemporary transformer model approaches.

In Chapter 7, we discuss the aforementioned results and highlight the advantages and disadvantages of our methods.

In Chapter 8, we conclude the work and provide suggestions for future work and improvements.

# Chapter 2

# Datasets

In this chapter, we describe the datasets used for this project. Annotated datasets specifically intended for GED and GEC of the Swedish language are sparse. To the best of our knowledge, only Swedish Learner Language (SweLL) and Dataset Linguistic Acceptability Judgements for Swedish (DaLAJ) exist, with DaLAJ being derived from SweLL. In this work, we use the DaLAJ dataset, as well as a raw corpus containing Swedish Wikipedia text entries. We artificially insert grammatical errors into the latter, to bolster the limited human-annotated data that exists.

## 2.1 DaLAJ v1.0

**Table 2.1:** The format of DaLAJ v1.0.

| Column | Example | Explanation |
|---|---|---|
| original sentence | För det andra är det nyckeln av livet. | |
| corrected sentence | För det andra är det nyckeln till livet. | |
| error indices | 29-30 | String indices of the error |
| corrected indices | 29-32 | String indices of the correction |
| error-corr pair | av-till | |
| error label | L-W | |
| L1 | Somaliska | Learner's mother tongue |
| Level | Intermediate | level of the course that learners takes |

Volodina et al. presented a dataset called DaLAJ in 2021. An extended version of the dataset, DaLAJ-GED, was later released by Volodina et al. (2023). Here, we refer to the original version of DaLAJ as DaLAJ v1.0. The sentences in DaLAJ v1.0 are derived from the SweLL dataset (Volodina et al., 2019), which is an error-annotated corpus comprising 502

essays written by second language learners of Swedish. 4,798 pairs of incorrect and correct sentences make up DaLAJ v1.0, and the sentences were manually altered by the creators to only contain one error each. Sentences in SweLL that originally contained multiple errors therefore occur several times in DaLAJ v1.0, each containing one of the original errors. Table 2.1 shows an example extract from DaLAJ v1.0, omitting the index and split columns.

The correction (column name: `error label` in the dataset) can be one of four possible types, and examples of each error type are shown in Table 2.2.

- `L-W`: Lexical-Wrong word. A wrong word or phrase in the text should be replaced.
- `L-Der`: Lexical-Derivation. Word formation (such as suffixes) should be corrected.
- `L-FL`: Lexical-Foreign language. A foreign word should be corrected to Swedish.
- `O-Comp`: Orthographic-Compounding. Spaces or hyphens between words should be removed or added.

**Table 2.2:** Examples of correction tags in DaLAJ v1.0.

| Error label | Original sentence | Corrected sentence |
|---|---|---|
| L-W | På kvällen går du till gym. | På kvällen går du på gym. |
| L-Der | Hur ska politikerna gå tillvägas? | Hur ska politikerna gå tillväga? |
| L-FL | De ligger på första plats i den league. | De ligger på första plats i den ligan. |
| O-Comp | Tillslut sov jag på soffan. | Till slut sov jag på soffan. |

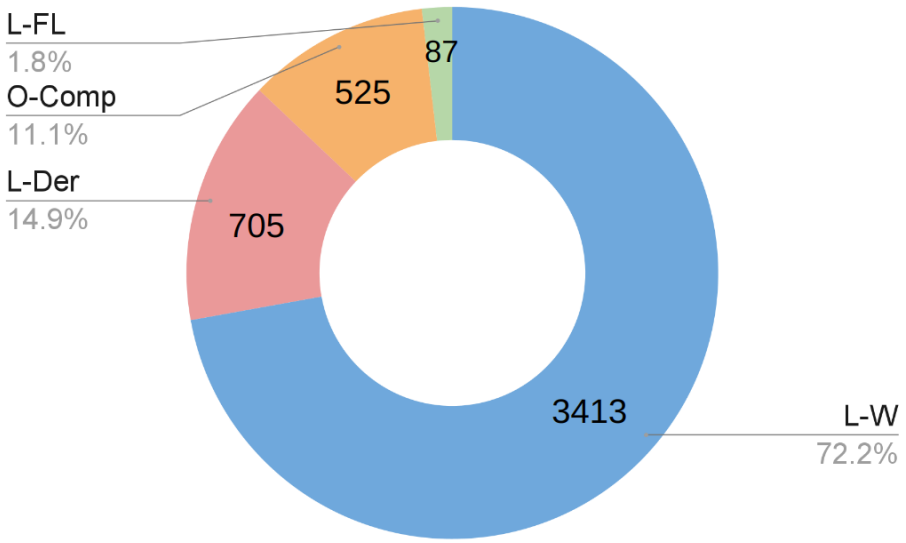


**Figure 2.1:** The distribution of correction tags in DaLAJ v1.0.

Figure 2.1 illustrates the distribution of the correction tags in DaLAJ v1.0. According to the figure, 72.2% of the correction tags in DaLAJ v1.0 are of type `L-W`. Note that the version of DaLAJ v1.0 that we acquired through Språkbanken[1] in January of 2023 contained 4,730 pairs of errorful and correct sentences.

[1]https://spraakbanken.gu.se/resurser

## 2.2 DaLAJ-GED

**Table 2.3:** Annotated datasets in Swedish.

| Corpus | Error Types | Sentences | Splits | Sentences |
|---|---|---|---|---|
| DaLAJ v1.0 | 4 | 9,460 (4,730 pairs) | Train | Incorrect: 3,841 |
| | | | Validation | Incorrect: 445 |
| | | | Test | Incorrect: 444 |
| DaLAJ-GED | 5 | 44,654 | Train | Correct: 17,472 |
| | | | | Incorrect: 18,109 |
| | | | Validation | Correct: 2,424 |
| | | | | Incorrect: 2,278 |
| | | | Test | Correct: 2,219 |
| | | | | Incorrect: 2,152 |

DaLAJ-GED is an extension of DaLAJ v1.0 by Volodina et al. (2023). It is made up of 44,654 sentences from the SweLL (Volodina et al., 2019) and Corpus of Coursebooks for Swedish as a Second Language (COCTAILL) (Volodina et al., 2014) corpora. It preserves the idea behind DaLAJ v1.0, namely that each sentence in the dataset should only contain a single error. Unlike its predecessor, DaLAJ-GED does not contain the corrected version of incorrect sentences. These can however be manually reconstructed using the confusion pair and error indices. As Table 2.3 shows, DaLAJ v1.0 contains 4,730 pairs of sentences and 9,460 sentences in total, and DaLAJ-GED is about five times larger than DaLAJ v1.0.

Table 2.4 shows an incorrect entry from DaLAJ-GED. Note that the city name has been pseudonymized to *"A-stad"*.

**Table 2.4:** The format of DaLAJ-GED.

| Column | Example | Explanation |
|---|---|---|
| Sentence | A-stad liger bland många fina berg. | |
| Label | incorrect | |
| Error span | 7-12 | |
| Confusion pair | liger-ligger | |
| Error label | O | |
| Education level | Nybörjare | Level of proficiency |
| L1 | Arabiska | Learner's mother tongue |
| Data source | SweLL | SweLL or COCTAILL |

The correction (column name: `Error label` in the dataset) can be one of five possible types, and examples of each error type are shown in Table 2.5.

- `P`: Punctuation should be moved/added/replaced.
- `O`: Orthographic. Spelling errors, upper/lower case and problem with compounding should be corrected.

- `L`: Lexical. Word formation (compounding), foreign word, wrong word or phrase should be corrected.
- `M`: Morphological. Morphological error such as verb/adjective/noun forms should be corrected.
- `S`: Syntactical. Word order or missing word should be corrected.

**Table 2.5:** Examples of the correction tags in DaLAJ-GED.

| Error label | Original sentence | Corrected sentence |
|---|---|---|
| P (Punctuation) | Hoppas vi ses snart | Hoppas vi ses snart. |
| O (Orthographic) | Annars är alt bra. | Annars är allt bra. |
| L (Lexical) | Vi flyttade i Sverige 2016. | Vi flyttade till Sverige 2016. |
| M (Morphological) | Alla mina dag gick så dåligt. | Alla mina dagar gick så dåligt. |
| S (Syntactical) | Jag bor ett hus. | Jag bor i ett hus. |

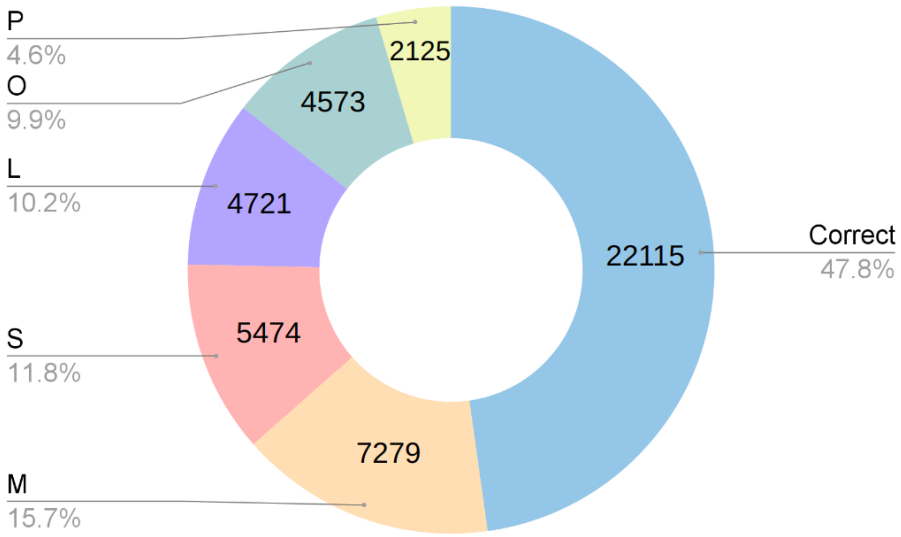


**Figure 2.2:** The distribution of error types in DaLAJ-GED.

Figure 2.2 illustrates the distribution of correction tags in DaLAJ-GED. Compared to DaLAJ v1.0 in Figure 2.1, the error types are more evenly distributed in DaLAJ-GED. Table 2.6 shows the most common errors of each error type.

Many sentences in DaLAJ-GED contain more than one error label. We used all sentences without duplication or removal for the tasks of binary sentence and binary token classification. For multi-label token classification, we duplicated the sentences containing more than one error type, and assigned one error label per such sentence. For example, a sentence originally labeled `L, M` would be transformed into two sentences, one labeled `L`, and the other `M`. Furthermore, we discarded the 10 entries that were missing error labels altogether. Therefore the total number of sentences in Figure 2.2 is 46,287, which exceeds the size of the original dataset.

**Table 2.6:** The most common errors in DaLAJ-GED. The third most common error in S-type ([inte]) involves incorrect word order errors.

| Error type | Rank | Error | Correction | Number of sentences |
|---|---|---|---|---|
| P | 1 | [ ] | [ , ] | 1312 |
| | 2 | [ ] | [ . ] | 314 |
| | 3 | [ , ] | [ . ] | 233 |
| S | 1 | [ ] | [att] | 450 |
| | 2 | [ ] | [det] | 212 |
| | 3 | [inte] | [inte] | 178 |
| M | 1 | [ ] | [en] | 347 |
| | 2 | [ ] | [ett] | 199 |
| | 3 | [en] | [ett] | 107 |
| L | 1 | [ i ] | [på] | 112 |
| | 2 | [ på ] | [i] | 93 |
| | 3 | [ till ] | [för] | 52 |
| O | 1 | [ jag ] | [Jag] | 61 |
| | 2 | [ det ] | [Det] | 58 |
| | 3 | [ har ] | [här] | 29 |

Figure 2.3 illustrates the number of words in the sentences in DaLAJ-GED. According to the figure 99% of the sentences contain fewer than 50 words. This information is used when we set the max number of words to be generated by models in Section 3.2.

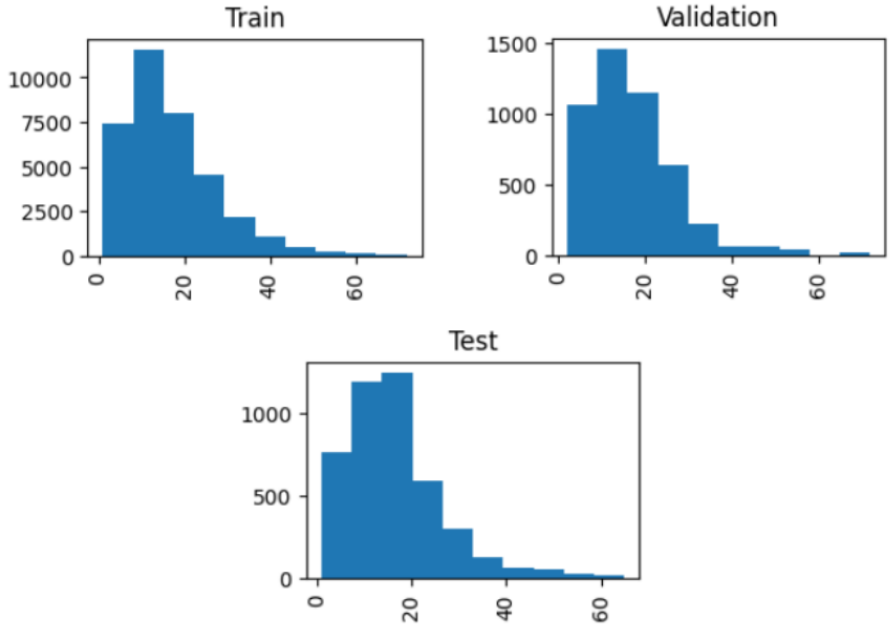


**Figure 2.3:** The number of words in sentences in DaLAJ-GED.

## 2.3 Swedish Wikipedia

Swedish Wikipedia is a corpus released by Gothenburg University's Språkbanken, which comprises 29,471,436 sentences scraped from a range of Swedish Wikipedia articles. It consists of mostly grammatically correct sentences of varying length. We used this raw corpus for the generation of sentences containing artificial grammatical errors. These artificially corrupted sentences were employed as extra training data for the models, in an attempt to boost performance. This is further described in Section 5.2.1.

# Chapter 3
# Previous Work

In this chapter, we present some previous works in the field of GED and GEC.

## 3.1 Grammatical Error Detection

Grammatical error detection (GED) is the task of detecting and potentially classifying the errors in a sentence.

The oldest and most traditional methods for GED and grammatical error correction (GEC) are rule-based. For instance, in 1980 IBM began developing the EPISTLE system to aid people in writing correct business letters (Miller, 1980). After parsing each word's part of speech, such as noun, verb and adjective, the EPISTLE system checked the input sentences against 250 rules designed specifically for English (Heidorn et al., 1982). In this manner, rule-based methods detect and correct errors using a parser and morphological rules. However, there are many exceptions to a limited ruleset, and therefore new methods were ultimately needed.

Statistical machine translation (SMT) is an approach to translating text from a source language into a target language, and neural machine translation (NMT) uses a neural network model to learn how to map source sentences to target sentences. These were first used for GED and GEC by posing the problem of translating incorrect sentences into correct sentences as a translation from an incorrect language into a correct language. These methods were extensively used for GEC tasks, and therefore both SMT and NMT are described in further detail in Section 3.2.

Accordingly, we will only briefly discuss some NMT-based methods for GED here. Rei and Yannakoudakis (2016) presented one of the first neural networks designed for GED of language learner writings. The authors investigated convolutional neural networks (CNNs), bidirectional recurrent neural networks (Bi-RNNs) and bidirectional long short-term memory networks (Bi-LSTMs) with series of tokens as input. Among these neural networks, Bi-LSTMs were found to perform best on the GED task. Rei and Søgaard (2018) extended the

Bi-LSTM model and proposed a model combining both token-level and sentence-level classification. This model outperformed Rei and Yannakoudakis (2016)'s Bi-LSTM and set new state-of-the-art results on the FCE, CoNLL-14 and JFLEG benchmarks.

Since the transformer-based BERT model was presented in 2018, attempts have been made to use pre-trained models for GED. Kaneko and Komachi (2019) explored applying extra attention mechanisms to each layer of BERT for a token-level GED task, and the resulting model outperformed both BERT and the model by Rei and Søgaard (2018) mentioned earlier, on FCE, JFLEG and CoNLL14-2. Yuan et al. (2021) implemented a binary classification model based on ELECTRA (another transformer-based model) and extended it to multi-label classification model for multi-label GED. This in turn outperformed Kaneko and Komachi (2019).

## 3.2 Grammatical Error Correction

Grammatical error correction (GEC) is the task of generating a grammatically correct sentence from an incorrect one.

As we mentioned in Section 3.1, statistical machine translation (SMT) translates a source sentence into a target sentence using statistics and probability. To compute the relevant probabilities, we first need a parallel corpus of incorrect and correct sentences. For example, if we have an incorrect word **scool**, we can calculate the conditional probability that **scool** is mapped to the correct word **school**. This is done using Bayes' theorem:

$$\mathrm{P}(\textbf{school}|\textbf{scool}) = \frac{\mathrm{P}(\textbf{school})\mathrm{P}(\textbf{scool}|\textbf{school})}{\mathrm{P}(\textbf{scool})}. \tag{3.1}$$

In this way, the conditional probability for other corrections such as P(**cool**|**scool**) can also be computed. The conditional probabilities for all candidate corrections are compared, and the most probable correction is selected. One advantage of using SMT models for GEC is that they do not require any updating of rules like rule-based models, making them easier to develop and maintain. However, the correction is made based on statistical information of individual words and it does not take the context of entire sentence into consideration.

Neural machine translation (NMT) also needs a parallel corpus of correct-incorrect sentences, but it uses neural networks to learn how to map an incorrect sentence to a correct sentence. Sutskever et al. (2014) presented a sequence-to-sequence learning model using a multilayered LSTM and Cho et al. (2014b) proposed an RNN encoder-decoder model. Both models consist of two RNNs called the encoder and decoder.

Figure 3.1 illustrates how encoder-decoder models can be adapted to GEC tasks. The orange-colored part is the encoder that reads the input sentence and computes the hidden states, while the blue-colored part is the decoder that generates the output sentence. An input sentence *"He drink water"* is inserted, and the model reads the sentence token by token. The hidden state of the encoder is updated as it reads each token of the input sentence and this continues until the model reads the end-of-sentence token. The decoder is trained to generate the next token given (1) the final hidden state of the encoder and (2) the previously generated token.

Since Sutskever et al. (2014) and Cho et al. (2014b) showed that their models outperformed SMT models for translation tasks, encoder-decoder models has been actively used.

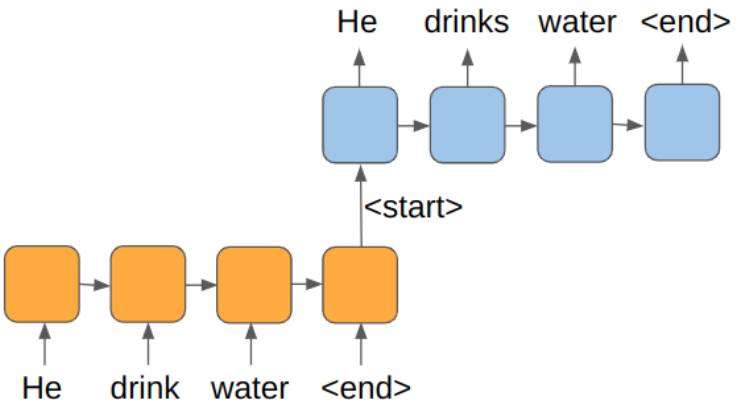


**Figure 3.1:** The illustration of encoder-decoder model for GEC tasks.

Specifically, Cho et al. (2014a) indicated that RNN encoder-decoder models perform well on short sentences, but perform worse as the length of the sentence grows. Subsequently, attempts were made to improve the performance on longer sentences. For example, Bahdanau et al. (2016) implemented a model using a bidirectional RNN encoder-decoder using scores, while (Kalchbrenner et al., 2017; Gehring et al., 2017) used an encoder-decoder model with CNNs. Finally, in 2017, Vaswani et al. proposed the transformer, a model that uses the attention mechanism of encoder-decoder models. This model outperformed the best previously reported models on the WMT 2014 English-to-German translation task, and many variants of the model have since been developed. In this thesis, we focus on transformer-based models which will be explained in detail in Section 4.

To conclude this section we introduce two current state-of-the-art GEC models.


**Figure 3.2:** An illustration of how the model by Rothe et al. (2021) is pre-trained and fine-tuned for the GEC task.

Rothe et al. (2021) present a way to train state-of-the-art GEC models by making use of pre-trained multilingual sequence-to-sequence models. Specifically, they propose a fully unsupervised method of pre-training the mT5 model's weights on a large raw multilingual corpus that has been split into sentences, 98% of which have been synthetically corrupted. The corruptions are simple and include dropping spans of tokens and characters, token and character swaps, character insertions and incorrectly lower-casing and upper-casing words. Two percent of the data is left uncorrupted. Finally the model is fine-tuned on human-annotated GEC data for each language of interest. Figure 3.2 illustrates how the model of Rothe et al. (2021) is pre-trained and fine-tuned. This model outperformed the previous state-of-the-art results on GEC benchmarks in English, Czech, German and Russian.

Omelianchuk et al. (2020) takes a different approach to GEC, by posing it as a sequence tagging task instead of a sequence generation task. This allows for faster inference times

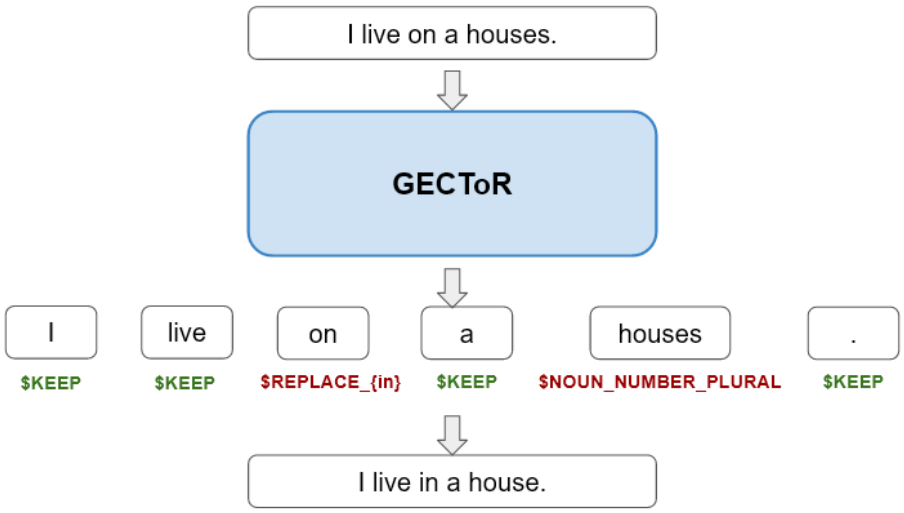


**Figure 3.3:** An illustration of how GECToR by Omelianchuk et al. (2020) performs sequence-tagging for the GEC task. The descriptive tags make it possible to reconstruct the correct sentence. If more changes are required, the output sentence can be tagged and altered again.

which can be crucial in live applications. Specifically, their system, called GECToR, involves training a model to label sequences with e.g. `KEEP`, `DELETE`, `APPEND` and `REPLACE` tags. For example, if the model predicts that a token should be replaced with another token $x$, it will label the original token with `REPLACE_{`$x$`}`. For this reason, a large error tag vocabulary is needed, and the authors use one of size 5000. Additionally, some token-independent tags are used; among others, `MERGE`, which merges the tagged token with its successor, and `SPLIT` which splits the current token into two new tokens. From these highly descriptive tags, a corrected sequence can be constructed. If desired, the corrected sequence can be tagged again, iteratively, until no more corrections are made. Figure 3.3 illustrates how the system operates.

## 3.3 GED and GEC for Swedish

In the 1990s, Lingsoft carried out research on GED and GEC for the Swedish language and Grammatifix (Arppe, 2000; Birn, 2000) was one of the early products. Grammatifix, which uses rule-based methods, was launched with Word 2000.

KTH Royal Institute of Technology initiated the Granska project in 1994 and presented a rule-based Swedish grammar checker called Granska (Carlberger et al., 2004). Figure 3.4 shows the structure of Granska. After tokenizing and part-of-speech tagging each sentence, they are checked by a rule matcher which generates correction suggestions. Stava, the spelling checker of Granska, is implemented with 1,000 suffix rules and the word lists from the Swedish Academy and Newspaper corpora (Kann et al., 1998). There are three different word lists:

1. a list of independent words that cannot be part of a compound,

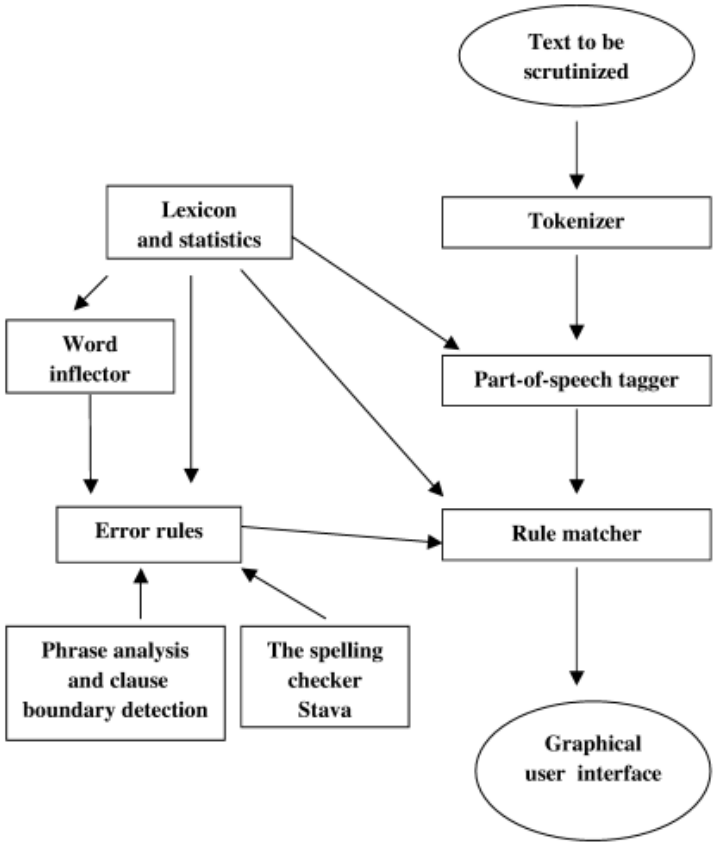


**Figure 3.4:** The overview of Granska system, image courtesy of Knutsson (2005).

2. a list of the words that can end a compound or be an independent word,

3. a list of the words that can be the first or middle part of a compound.

When an input word is inserted, the program checks if the word belongs to list 1) or 2). If not, the word is reconstructed by suffix rules and the program again checks if the constructed words belong to the lists.

Kann et al. (1998) described how Granska looks up the word *poslinsdockorna* in its lists to check the spelling. Since the word does not exist in list 1) and 2), the program breaks the word into two parts. The first part of the compound, *porslins-* exists in list 3), but the last part of the compound, *dockorna* are still not found in any of the lists.

The suffix rule that is consulted in this case is:

$$\text{-orna} \longleftarrow \text{-a, -an, -or}$$

Because the reconstructed words, *docka*, *dockan* and *dockor*, are found in the list 1) or 2), *dockorna* is considered as a legal word, and the word *poslinsdockorna* is also correct word according to Granska's spell checker.

There were also some attempts to use statistical and neural machine translation for Swedish GEC. Bigert (2002) presented a probabilistic method for GED, Probgranska, based on the statistics of a corpus containing correct sentences. Probgranska checks how often the part of

speech and form of a word follow each other, and based on this, outputs if the word is correct or incorrect (KTH Royal Institute of Technology, 2020).

Stymne and Ahrenberg (2010) presented an SMT model for Swedish GED. They improved the performance of the SMT model by combining it with Granska, whereby Granska evaluates the outputs from the SMT models. Snålgranska (Sjöbergh and Knutsson, 2005) is a GED model using machine learning trained on a corpus with errors.

More recently, some transformer-based models trained on Swedish corpora have been presented. For example, the National Library of Sweden has released additional versions of the BERT and BART models since they originally presented KB-BERT (Malmsten et al., 2020). In 2023, AI Sweden released GPT-SW3 (Ekgren et al., 2022) which is based on the GPT architecture. In the field of GED, Volodina et al. (2021) implemented and evaluated an LSTM model fed with KB-BERT embeddings as input data, and Volodina et al. (2023) fine-tuned KB-BERT on DaLAJ-GED. These experiments were conducted to evaluate the newly created DaLAJ datasets, and demonstrating the potential for fine-tuning transformer-models for GED and GEC tasks in Swedish. Further details on this topic are provided in Section 4.1.2.

## 3.4 Synthetic Data

The recent advances in GED and GEC using neural architectures typically require large parallel training data of incorrect and correct text. Such human-annotated data is scarce, especially for a relatively low-resource language such as Swedish. For this reason, it is natural to look to synthetic data. Several attempts to use synthetic data to improve model performance have been done. Indeed, several state-of-the-art models make use of vast amounts of synthetic training data. For example, the methods described by Rothe et al. (2021) and Omelianchuk et al. (2020) that were introduced in Section 3.2.

What defines good synthetic data for GED and GEC is that it accurately reflects the broad distribution of errors made by humans. Stahlberg and Kumar (2021) propose a novel way of generating synthetic data that more closely aligns with the range of grammatical errors made by humans, and indeed show that systems trained on this data can even surpass state-of-the-art systems trained on human-annotated data. Their method involves generating an incorrect sentence from a correct sentence given an error label from a set of 25 error labels supported by the toolkit ERRANT (Felice et al., 2016; Bryant et al., 2017). The goal is then to assign a single error tag to each sentence in the training data such that the distribution of error tags matches that which is found in real data.

Casademont Moner and Volodina (2022) study the distribution of error types present in authentic second language learner data, specifically the SweLL corpus (Volodina et al., 2019), from which the DaLAJ-GED corpus is derived. They report what the most common error types are and develop a corruption pipeline which inserts artificial errors into grammatical sentences. They find that an addition of some synthetic data during the training procedure can improve model performance, but that too much synthetic data of a specific error type will hurt the model's ability to classify other error types.

## 3.5 Sentence Scoring

In the context of GEC, the likelihood of a sentence can be of interest as a measure of how correct it is. In the context of language models like BERT or GPT, one can make the assumption that the models have been trained on mostly grammatically correct text, and will therefore assign a higher probability to a grammatically correct sentence than an ungrammatical one.

Research in this field has employed both masked language models (MLM) such as BERT (Salazar et al., 2020) and causal language models (CLM) such as GPT (Weng et al., 2020). MLMs have the benefit of being able to incorporate a bidirectional context when evaluating the probability of each token, whereas the CLMs benefit from only requiring a single forward-pass to evaluate a whole sentence, unlike MLMs which require one pass per token in the sentence. Attempts have been made to utilize the beneficial characteristics of both types of language models, one example being the sliding language model by Song et al. (2022).

Furthermore, works like LM-Critic by Yasunaga et al. (2021) impose a local-neighborhood criterion which means that, when evaluating a number of proposed corrections of a sentence, the corrections are required to be within edit-distance 1 of the original, incorrect sentence. LM-Critic uses language modeling to select the best, or most likely corrected sentence, among a number of candidates that are all in the local neighborhood of the sentence that needs to be correct.

# Chapter 4
# Transformers

In this thesis, we explore transformer-based models for GED and GEC. This chapter describes the transformer architecture and its application for this work. Moreover, this chapter provides knowledge about how text and words are represented by the models, by tokenization and embeddings.

## 4.1 Models

Common to most popular machine learning models used today for GED and GEC is their immense size, and that they have been trained on vast amounts of data. These factors alone make it difficult to train these models from scratch on consumer-grade hardware. Fortunately, there exist many free and open libraries which provide pre-trained models for all to use. Typically these models have been trained to learn an internal representation of natural language, also called a language model. This language model can be used to great advantage when fine-tuning on various specific tasks, such as question-answering or sentence classification.

Fine-tuning is the process of using a pre-trained model for a specific downstream task. This involves replacing the output layer of a pre-trained model with a new head. Figure 4.1 illustrates how we fine-tune a BERT model for example. The head is typically a small neural network whose purpose is to be trained on a specific task. Thus, the input data first passes through the large pre-trained model, followed by the small neural network which produces the final output. This way, the complex language model of the pre-trained network can be leveraged to great effect on a wide range of tasks. When training the new head, the parameters of the pre-trained network can be frozen, meaning that they do not change. The purpose of this is to preserve the learned knowledge of the pre-trained model. Alternatively some of the final layers of the pre-trained model can be unfrozen and trained together with the new head. In this thesis we investigate both approaches, for different tasks.

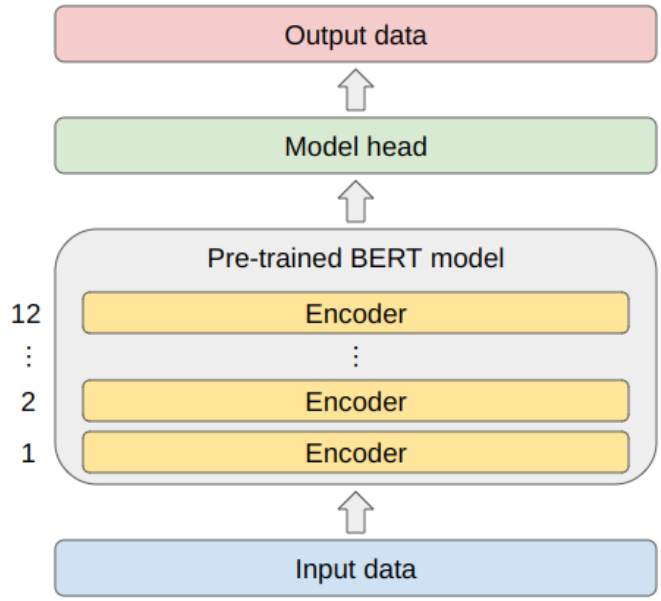


**Figure 4.1:** An illustration of the components of a fine-tuned BERT model.

### 4.1.1 Transformers

In 2017, Vaswani et al. introduced the transformer, a type of feedforward neural network based on the concept of attention. In the years that followed, significant advances were made in the field of GED and GEC using transformer-models (Junczys-Dowmunt et al., 2018). The transformer, which uses an encoder-decoder architecture, enabled the training of models on large data to create language models without the need for annotated training data. Through the act of observing vast amounts of written text, these transformer-models can form an internal model of language that can be leveraged for a wide range of tasks, from e.g. machine translation to GEC or text generation. The transformer, with its encoder and decoder, was originally designed for sequence-to-sequence tasks such as translation between languages, but shortly after its release, different models that made use of solely the encoder or the decoder were published. We describe some of these models in Section 4.1.2.

**Attention** The key feature of the transformer is its heavy reliance on the attention mechanism. This mechanism allows a neural network to give different weight or *"attention"* to each part of a sequence. In practice, it is a mapping of key-value pairs together with a query, to an output, all of which are vectors. The query may be the token embedding we wish to calculate the attention for, and the keys may be all other tokens embeddings in the context around the query token. In reality, multiple queries are processed at once, and thus we can stack the queries, keys and values into matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$. We derive a query by multiplying a query token with the matrix $\mathbf{Q}$. Similarly, keys are obtained by multiplying each key token with the matrix $\mathbf{K}$, and the values by multiplying the key tokens with the matrix $\mathbf{V}$. The attention scores for our query token are then calculated by taking the dot product between the keys and the query. Queries and keys that are similar will yield a large dot product and thus large attention scores. These scores are then scaled with a factor $\sqrt{d_k}$ where $d_k$ is the

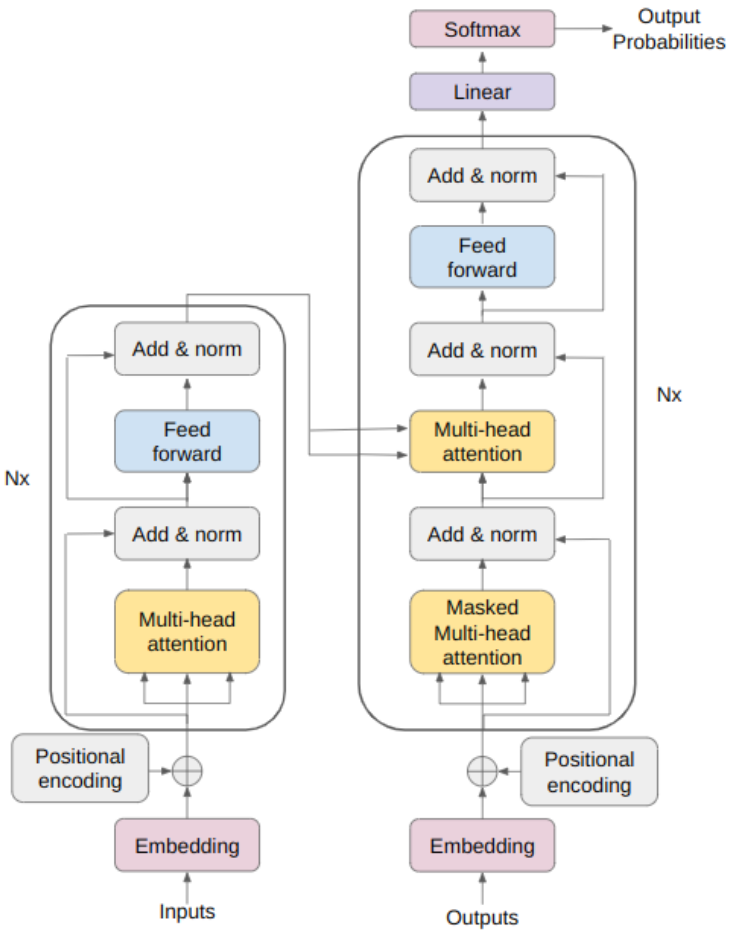


**Figure 4.2:** The transformer architecture. The block on the left is the encoder, and the block on the right is the decoder. Adapted from Vaswani et al. (2017).

dimension of each key and query vector, to ensure that the attention scores all have variance 1. Additionally, they are normalized using a softmax function. Finally, the output is obtained by multiplying the attention scores with the values. Equation 4.1 shows the derivation of the attention.

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{4.1}$$

**Self-Attention** With self-attention, the keys, values and query are all derived from the same words, e.g. from the same written sentence. In other words, self-attention computes how the parts of a sequence relate to each other. This mechanism allows the transformer to process the whole context around each token without the need for a recurrent network structure. Notably this enables parallel processing of tokens.

**Multi-Head Attention** In order to create the key, query and value vectors, linear transformations are applied to each input embedding. The parameters of these transformations are not set by the programmer, but learned from the data, together with the other parameters of the model. Multi-head attention means having more than one set of transformations for generating the key, query and value vectors. Each set of transformations is referred to as one attention head, and each head may learn to attend to different aspects of the tokens in the input sequences, such as how different parts of speech (PoS) interact with one another.

**The Encoder** In the context of transformers, the encoder typically consists of several stacked encoder layers. This is illustrated in the left half of Figure 4.2, where $N$ encoder layers are denoted by $N\times$. The input to each encoder layer is a sequence of embeddings, to which self-attention is applied, after which they are each fed through a fully connected feedforward neural network. The outputs of each encoder layer is of the same size as the inputs, so that it may be fed into the next encoder layer. As the embeddings pass through the encoder stack, they will become increasingly contextualized.

**The Decoder** The transformer's decoder consists of several stacked decoder layers, each of which contains two attention layers. The decoder with its $N$ decoder layers are illustrated in the right half of Figure 4.2, denoted by $N\times$. Given that the task of the decoder is to perform next-token prediction, the masked multi-head attention attention layer masks tokens in the right context, so that they cannot be used when predicting the current token. This is necessary to properly train the decoder on the task of next-token prediction, as knowing the right context would correspond to cheating. The other multi-head attention layer performs multi-headed attention on the key and value vectors output by the encoder, using the decoder's current representations of the sequence as queries. In other words, the decoder learns the relations between the representations of the two different sequences, the first one being the encoder's output and the other being the decoder's own output from the previous step. This is how the decoder can learn mappings between sentences of two different languages.

**Beam Search** As the decoder generates the next token, it selects the token with the highest probability at each step by default. This is not always desirable, as greedily selecting the most probable token at each step will not always yield the most probable sequence overall.

Beam search involves keeping track of the top-$n$ most probable next tokens, where $n$ is called the number of *beams*. This creates $n$ possible paths for the next token, i.e. $n$ branches of a search tree. This step is iterated for each branch until we reach a pre-determined maximum sentence length, or the end of the sentence. Finally, we select the path of the search tree by ranking the beams according to their log probabilities. Figure 4.3 illustrates the process.

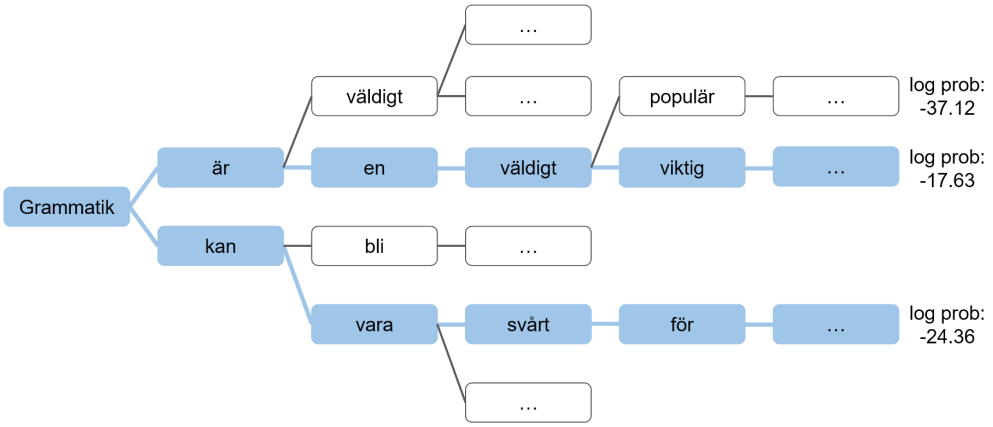


**Figure 4.3:** An illustration of beam search with two beams. For brevity, not all beams are drawn. The blue color signifies the two most likely candidate sequences. The candidate sequence with the highest log probability score is chosen as the output.

### 4.1.2 Pre-trained models

As previously mentioned, the computational resources, memory and time needed to train modern transformer-based language models means that it is not feasible to train them from scratch for each specific use-case. Instead, it is common practice to use the outputs from pre-trained language models as inputs to a neural network that is trained on a specific downstream task.

**BERT.** Devlin et al. (2019) introduced BERT (Bidirectional Encoder Representations from Transformers), which consists of multiple layers of bidirectional transformer encoders. BERT is pre-trained with two unsupervised tasks: masked language modeling and next sentence prediction (NSP). Masked language modeling (MLM) is a procedure whereby 15% of all tokens in each sequence are randomly masked and then predicted based on the surrounding context. The NSP is a task that involves letting the model predict whether two sentences follow each other or not. Devlin et al. (2019) showed that this task was beneficial to tasks such as question answering.

As we briefly described in Section 4.1, the pre-trained BERT model can be fine-tuned by adding a small number of additional layers on top, called the *head*, and training the head on a dataset specific to a task. Figure 4.4 illustrates how BERT can be fine-tuned for different tasks. For instance, example (a) in Figure 4.4 illustrates that BERT can be fed a sentence and return an output corresponding to a classification of the sentence. For example, the sentence can be categorized as correct or incorrect. Additionally, BERT can be fine-tuned for token-tagging as shown in (b) of Figure 4.4, enabling the model to return a sequence of labels, one for each token in the sentence. This makes it possible to fine-tune the model on the task of tagging each token as correct or incorrect, or apply more specific error tags to each token.

**mBERT.** In 2019, Google Research released a multilingual extension of BERT, which was conceptually identical to BERT. The 100 languages with the largest Wikipedia entries were used for training, and the model was trained on the entire Wikipedia for each language.

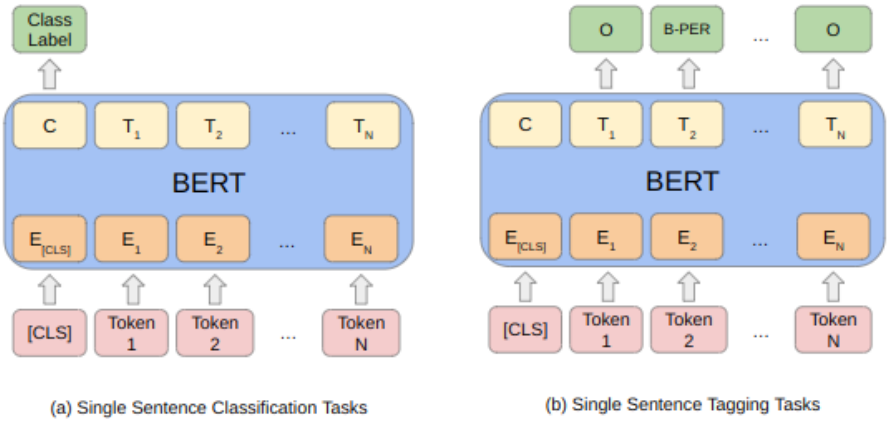


**Figure 4.4:** An illustration of how BERT can be fine-tuned on different tasks. Adapted from Devlin et al. (2019).

The vocabulary size was increased to 110K shared WordPiece embeddings, from the original 30K WordPiece embeddings. Google Research (2019) found that the multilingual BERT performed slightly worse than the monolingual BERT on high-resource languages such as English or Chinese, but that for low-resource languages, the effect of transfer-learning allowed the model to outperform models trained only on data from the low-resource language.

**RoBERTa.** Liu et al. (2019) argued that BERT was undertrained, and to remedy this the authors provided a modified BERT which they called RoBERTa. One of the key differences to BERT is that RoBERTa does not perform NSP in the pre-training stage, and that RoBERTa generates the masking pattern every time a sequence is fed to the model whereas BERT uses a static masking. Additionally, the model was trained with approximately 10 times more data as well as longer input sequences. As RoBERTa does not have NSP as a training objective, the inputs are no longer sentence pairs, but full sentences with a maximum length of 512 tokens. The authors found that RoBERTa outperformed BERT using both the GLEU and SQuAD evaluation metrics.

**XLM.** Following the success of BERT, which is based on the encoder part of a transformer, and GPT, which is based on the decoder part of a transformer, Lample and Conneau (2019) presented a transformer intended to work for multiple languages. This required training a model on data from multiple languages, and the training objectives were similar to the ones used by both BERT and GPT, as well as a third objective, specifically targeting multiple languages. This type of model was dubbed a Cross-lingual Language Model or XLM, and its three training objectives were:

1. Causal Language Modeling (CLM) relating to predicting the next token based on the previous tokens;

2. Masked Language Modeling (MLM) involving predicting a masked token based on both the left- and right contexts and

3. Translation Language Modeling (TLM) which involves mixing sentences of different languages, masking tokens, and predicting the masked tokens based on the left- and

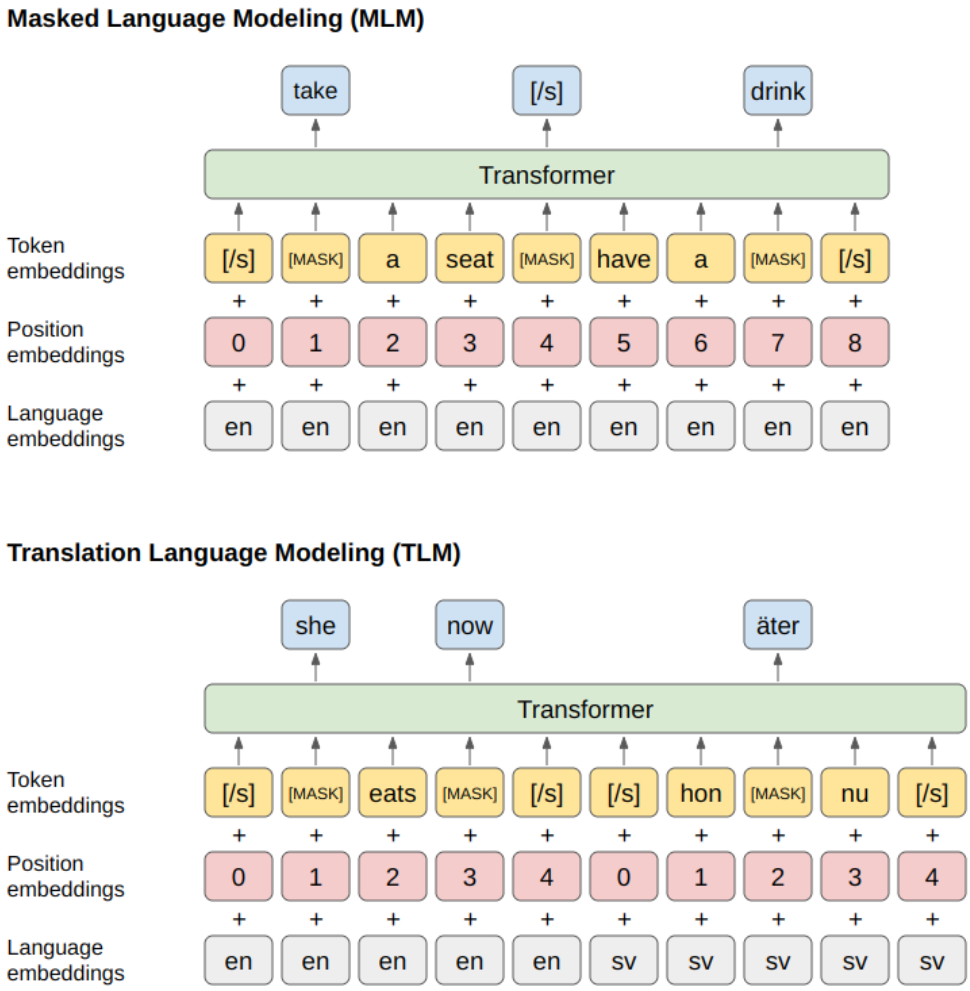


**Figure 4.5:** Masked Language Modeling (MLM) and Translation Language Modeling (TLM). Adapted from Lample and Conneau (2019).

right contexts (which may be in different languages).

As a result of training on multiple languages, the XLM was able to produce similar word embeddings for the same word from multiple languages. For example, the word embeddings for the English word 'cat' and the Italian word 'gatto' were similar. Lample and Conneau (2019) found that the model was able to reduce perplexity for low-resource languages like Nepali, by including training data from other languages such as Hindi and English. A model trained on Nepali, Hindi and English was thus better at predicting Nepali, than a model that was solely trained on Nepali.

**XLM-RoBERTa.** Inspired by the pre-training methods of RoBERTa, Conneau et al. (2019) attempted to tune the concept of XLM by following suit and dropping the NSP training objective, as well as using the dynamic masking scheme from RoBERTa. Additionally, unlike XLM, XLM-RoBERTa does not use language embeddings, which aids the model in alternating between different languages.

The authors found that adding more languages to the training data improved model performance to a point of approximately 15 languages, especially for low-resource languages. However, adding even more languages was found to degrade model performance, due to the so called *curse of multilinguality*. As more languages are added, less of the model's capacity is available to learn representations for each language. This problem was partly alleviated in XLM-RoBERTa by increasing the size of the hidden representations and increasing the vocabulary size. As a result, XLM-RoBERTa produced state-of-the-art results on several language tasks, especially for low-resource languages.

**Swedish BERT models.** There have been several projects dedicated to creating a Swedish version of BERT. The Swedish Public Employment Service developed SweBERT (Swedish Public Employment Service, 2020) by training the model on Swedish Wikipedia data consisting of approximately 2 million articles. The same year, the National Library of Sweden (KB) developed a Swedish BERT (KB-BERT) by training it on newspapers, official reports by governments, legal e-deposits, social media as well as the Swedish Wikipedia (Malmsten et al., 2020).

Malmsten et al. (2020) compared the performances of SweBERT and KB-BERT with mBERT, by evaluating the models on named entity recognition (NER) and part-of-speech tagging tasks. The results showed that KB-BERT outperformed mBERT and SweBERT for NER tagging.

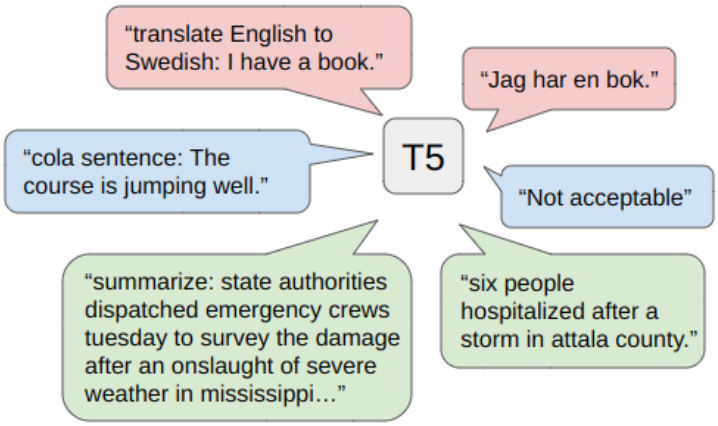


**Figure 4.6:** A diagram depicting T5 and its multiple training objectives. Adapted from Raffel et al. (2019).

**mT5.** Multilingual T5 (mT5) by Xue et al. (2021) is a transformer model using the encoder-decoder architecture. It is a multilingual extension of T5 (Raffel et al., 2019), trained using the same training objectives as T5 on corpora covering 101 languages.

The T5 model, and consequently mT5, are pre-trained sequence-to-sequence models that can be used for a wide range of downstream tasks. Their sequence-to-sequence nature means that they generate output text when given an input sequence of text. As shown in Figure 4.6, they can be used for tasks such as machine translation, text summarization and classification. In the classification case, the models can be trained to output the name of the predicted class.

The models' training objective is masked language modeling, specifically *"span-corruption"*, where a span of consecutive tokens is masked, and the model is tasked with predicting the masked tokens.

**GPT.** Generative Pre-trained Transformer (GPT) by Radford et al. (2018) is a model based on several layers of transformer decoders. Given a sequence of input tokens, GPT uses multi-headed self-attention over the input context, feeds the results through a feedforward neural net and produces an output distribution over possible subsequent tokens. From this distribution, the most likely token can be sampled. This unsupervised pre-training was carried out on BooksCorpus, a large unlabeled corpus comprising 11,038 unpublished books. Specifically, GPT was trained on lengthy contiguous pieces of text, which allows the model to take very large contexts into account during its predictions.

GPT can be fine-tuned on a number of downstream tasks, such as classification, textual entailment, question answering and commonsense reasoning.

**GPT-SW3.** GPT-SW3 is a generative decoder-based model based on the GPT architecture (Radford et al., 2018) developed by Ekgren et al. (2022). It is trained on a novel 100 GB Swedish corpus consisting largely of web text data from discussion forums and news articles, and to a lesser extent literature, subtitles and wiki data, among other things. Like other models of the GPT family, its training objective is next-token prediction. GPT-SW3 is released in various sizes ranging from 126 million to 20 billion parameters, with more to come. The base 3.5 billion parameters model was shown to have lower perplexity than GPT2-XL (Radford et al., 2019), Flashback-GPT (Norlund and Stenbom, 2021) and GPT-Neo (Black et al., 2021) on Swedish corpora. Like GPT, it can be fine-tuned on various downstream tasks.

## 4.2 Text and Word Representation

### 4.2.1 Tokenization

In order for a machine-learning model to be able to process text, the text must first be converted into a language the model can understand. This typically involves converting characters, words or sub-words into numbers. How many unique numbers are used is referred to as the vocabulary size.

#### Word Tokenization

There are many ways of translating words into numbers, and in each case a selection must be done in order to restrict the vocabulary size for computational reasons. Table 4.1 shows how the words of a sentence can be converted into tokens, each represented by a numeric ID.

**Table 4.1:** A fabricated illustration of word tokenization of a sentence.

| **Words** | Tokenization | is | a | task | in | Natural | Language | Processing. |
|---|---|---|---|---|---|---|---|---|
| **Token IDs** | 7567 | 254 | 37 | 678 | 86 | 476 | 768 | 3258 |

Encoding each conceivable word with a unique number is unfeasible, as the number of model parameters rapidly increases with vocabulary size. A hypothetical scenario with 1 million unique words, and model input vectors of dimension 1,000, would result in $10^6 \cdot 10^3 = 10^9$ parameters just for the input mapping, which is on the scale of some of the large transformer models on the market.

For this reason, several methods have been proposed to instead encode sub-words.

### Sub-word Tokenization

Sub-word are units made up of one or several characters which can be combined to form whole words. The benefit of using sub-words is that the same sub-word can be used flexibly in the formation of many different words. For example, the words *great*, *greater*, *greatest* need not occupy one token ID each. Instead, they can be composed of the three sub-words: the common stem *great* and the two suffixes *er* and *est*. This way, the limited vocabulary size can be put to better use, as many sub-words can be used in the formation of many words. The vocabulary of sub-words is usually learned from a large corpus of data.

A popular method of sub-word tokenization is WordPiece (Schuster and Nakajima, 2012). WordPiece works by initializing a word inventory of elementary characters and symbols. The algorithm then involves forming new words by merging two units out of the word inventory. The newly formed word is the one that maximizes the likelihood on the training corpus, when it is added to a language model trained on the corpus. This is iterated until a predefined vocabulary size has been reached, or the likelihood no longer increases sufficiently.

Table 4.2 illustrates the result of sub-word tokenization of a sentence. The inserted prefixes **##** signify that the previous character is not a whitespace, and hence the two tokens should be merged when combining the tokens back into a string.

**Table 4.2:** A fabricated illustration of sub-word tokenization of a sentence.

| **Sub-words** | Token | ##ization | is | a | task | in | Natural | Language | Process | ##ing. |
|---|---|---|---|---|---|---|---|---|---|---|
| **Token IDs** | 4565 | 12389 | 337 | 276 | 136 | 2476 | 1365 | 4852 | 1239 | 326 |

## 4.2.2 Word vectors and Embeddings

### One-hot Encoding

Embeddings lend a way to work with tokens using numerical operations, whether those tokens are whole sentences, words, or sub-words. This is typically done by converting each token into a vector. A simple yet widely used example is one-hot encoding, which involves converting each token into a $d$-dimensional vector, where $d$ is the size of the vocabulary. Each token is represented as a vector consisting of zeroes in all dimensions except for one. As a result, the vector representations of the vocabulary are all equidistant and orthogonal to one another. One-hot encoding results in a very large and sparse vector-space.

## TF-IDF Vectors

Term Frequency-Inverse Document Frequency (TF-IDF) uses statistics to represent a word's importance in a given document. In other words, it describes a document in terms of what words are important to the document. If a word occurs frequently in a specific document, but infrequently overall, it is considered important to the document. To compute the TF-IDF score of a word, first the TF($t, d$)-part is calculated. This is the number of times the word ($t$) occurs in a document ($d$). Then, DF($t$), which is the number of documents ($N$) in which the word occurs, is also calculated. The IDF (Inverse DF) is then calculated as follows:

$$\text{IDF}(t) = \log(\frac{N}{1 + DF(t)}).$$

TF-IDF($t, d$) is obtained by multiplying TF($t, d$) and IDF($t$). In other words, TF represents how frequently a word appears in a document and IDF reduces the weight of very frequent words, so that TF-IDF gives less biased information about how important a word is.

A TF-IDF vector is a vector containing the TF-IDF scores of the words in a document, and it can be used as a vector representation of textual data when training machine learning models. In our case, each sentence in the dataset is considered a document.

## Embeddings

A downside to one-hot encoding is that the vector-space is large and sparse. For this reason, there have been several attempts to reduce the dimensionality of the vector representations. One successful example is Word2Vec authored by Mikolov et al. (2013), which involved learning efficient real-valued vector representations of words from a corpus of 1.6 billion words. The results are 300-dimensional vectors which capture both syntactic and semantic meaning. For example, Mikolov et al. (2013) found that the vector describing the word *Queen*, could be obtained by:

$$\text{vector}(Queen) = \text{vector}(King) - \text{vector}(Man) + \text{vector}(Woman).$$

These vectors were created for the 30,000 most frequent words in the corpus. This suggests that both syntactic and semantic information about 30,000 words has been captured in just 300 dimensions.

**Non-contextual Embeddings** Both Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), two successful attempts at word-vector representations, are referred to as non-contextual embeddings. Non-contextual meaning that the word vectors for a certain word do not change depending on how the word is used in a context.

**Contextual Embeddings** While the Word2Vec and GloVe embeddings led to new state-of-the-art results on many language tasks, they were still lacking a key component when it came to describing words: context.

Contextual embeddings are as the name suggests, contextual. Using the word *bank* as an example, we can imagine different uses for it in a sentence, such as "*... financial bank ...*" and "*... river bank ...*".

As humans, we know that these refer to two different types of *"bank"*, thanks to the context. This difference is however not captured by the Word2Vec or GloVe vector representations.

Recurrent neural networks (RNNs) enable the creation of embeddings which incorporate the context that precedes a word, into the word's vector representation. This means that for the word *bank*, we could have two different vectors for the word in the two previously described contexts.

**Bi-directional Contextual Embeddings** The architecture of RNNs meant that they were unable to fully incorporate the context on both sides of a word in a sentence. For example, again considering the word *"bank"* in the two contexts: *"... river bank ..."* and *"... the bank of the river ..."*. In this case, we may want the word *"bank"* to be represented by the same vector, as they refer to the same type of bank. In order to disambiguate *"bank"* from the financial institution in this case, we must consider the context to the right as well as to the left of the word.

The introduction of the transformer by Vaswani et al. (2017), which is described in Section 4.1.1 meant that embeddings which incorporated the context in both directions could be produced. Derivative transformer-models such as BERT were trained on large data to learn to produce such embeddings.

# Chapter 5
# Method

In this chapter, we describe how the models have been used and how the experiments have been carried out.

Broadly speaking, our work has been divided into three phases. The first involved training models to predict whether a sentence was grammatically correct or incorrect. The second phase involved training models to predict whether each token in a sentence was correct or incorrect. More specifically, we further trained models to predict what type of error, if any, each incorrect token corresponded to. Lastly, the third phase involved training encoder-decoder and decoder-based models to produce grammatically correct sentences from incorrect sentences.

## 5.1 Grammatical Error Detection

The experiments with grammatical error detection (GED) were conducted using pre-trained encoder-based models that were fine-tuned on the GED task. We perform baseline experiments using a traditional logistic regression for later comparisons with the transformer-based models. The pre-trained models were acquired from Hugging Face[1], a platform for sharing pre-trained machine learning models and datasets for e.g. natural language processing.

### 5.1.1 Binary Sentence Classification

We posed binary sentence classification as the task of classifying sentences as correct or incorrect. As Figure 5.1 illustrates, a model trained to perform binary sentence classification reads a tokenized sentence and outputs a label. Both correct and incorrect sentences of the datasets DaLAJ v1.0 and DaLAJ-GED were used to train the models for binary sentence classification, and the datasets contain labels 'Correct' and 'Incorrect' for each entry.

[1] https://huggingface.co

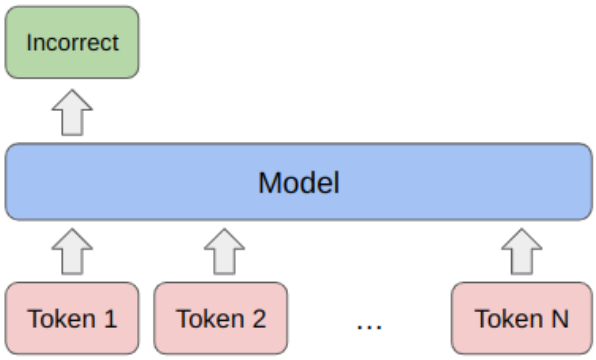


**Figure 5.1:** Diagram illustrating binary sentence classification.

**Logistic Regression.** Before conducting experiments with encoder-based models, we performed an experiment with a traditional logistic regression classifier to serve as a baseline for comparisons with further experiments. We created TF-IDF vectors from DaLAJ v1.0 as described in Section 4.2.2, and trained a logistic regression classifier on the TF-IDF vectors to predict whether each sentence from the test set was correct or incorrect. We compared the performance of the model with that of a model trained in the same way on the larger dataset DaLAJ-GED.

Inspired by Volodina et al. (2021)'s experiment, which used BERT embeddings instead of TF-IDF vectors in order to train a bidirectional long short-term memory (Bi-LSTM) classifier, we also trained a logistic regression classifier using BERT embeddings. To generate BERT embeddings, we used the KB-BERT model available on the Hugging Face platform (`KB/bert-base-swedish-cased`).

Specifically, we used the [CLS] embedding obtained from the final hidden layer of KB-BERT as features. The [CLS] embedding is the first embedding output by BERT, as seen in Figure 4.4, where it is denoted **C** for class label. This embedding is commonly used for sentence classification tasks as it is a contextualized representation of the sequence as a whole. We compared the performance of the logistic regression model trained on KB-BERT embeddings extracted from DaLAJ v1.0 with that of a model trained in the same way on KB-BERT embeddings extracted from DaLAJ-GED.

**Encoder models.** In addition to the logistic regression classifier, we directly fine-tuned several encoder-based models to classify sentences as correct and incorrect. This was done by using pre-trained models and adding a dropout layer followed by a single dense layer on top of the [CLS] embedding, denoted **C** for class label in (a) of Figure 4.4. Following the dense layer, there is a softmax classifier that outputs the probability of each class. By selecting the class with the highest probability, we can assign a class to each sentence. In our case, there were only two classes, correct and incorrect. We fine-tuned BERT and RoBERTa, as well as their multilingual counterparts, mBERT and XLM-RoBERTa. Lastly, we used a Swedish BERT model, KB-BERT. For all models, we froze their ten first layers and trained them with a learning rate of $10^{-5}$ for ten epochs with a batch size of 32. Table 5.1 lists the specific models used for binary sentence classification, and their respective designations in this thesis.

**Table 5.1:** The encoder models used for binary sentence classification.

| Model designation | Technical model name |
|---|---|
| BERT | `bert-base-cased` |
| RoBERTa | `roberta-base` |
| XLM-RoBERTa | `xlm-roberta-base` |
| mBERT | `bert-base-multilingual-uncased` |
| KB-BERT | `bert-base-swedish-cased` |

### 5.1.2 Binary Token Classification

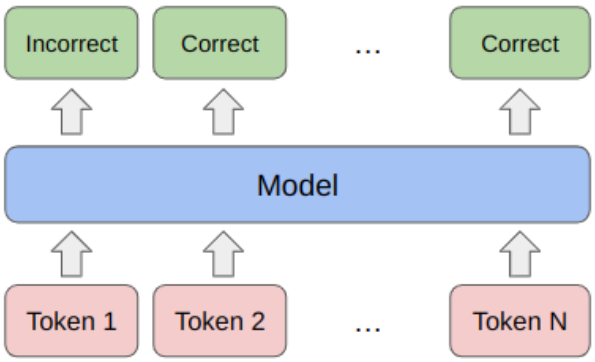


**Figure 5.2:** A diagram of binary token classification.

We pose binary token classification as the task of classifying each word or token in a sentence as either correct or incorrect. As illustrated by Figure 5.2, the model returns labels for each token in the input sentence. We used the sentences from the training set of DaLAJ-GED to train the models, and we labeled the tokens of each sentence in the training data with correct or incorrect by using the error indices. Thus, each sentence is associated with a sequence of labels, e.g. `[Correct, Correct, Incorrect, ... , Correct]`, which are used during the training procedure.

As can be seen in Section 6.1.1, our initial experiments with binary sentence classification showed that monolingual English models, such as BERT and RoBERTa, performed poorly on the task in Swedish. Therefore, for the subsequent classification tasks, we fine-tuned only the three best-performing encoder models: mBERT, KB-BERT and XLM-RoBERTa. Additionally, we observed that the results of the models trained on DaLAJ v1.0 were worse than those trained on DaLAJ-GED. Therefore, we used only DaLAJ-GED dataset for the experiments that followed.

The hyperparameters used for the models are shown in Table 5.2.

### 5.1.3 Multi-label Token Classification

We pose multi-label token classification as the task of classifying each token in a sentence as one of the five error types in DaLAJ-GED, described in Section 2.2, as illustrated by in Figure 5.3. The tokens of each sentence in the training dataset is associated with a sequence of labels

**Table 5.2:** Hyperparameters used for training the models on binary token classification.

| Model | Learning Rate | Epochs | Batch Size | Nbr of frozen layers |
|---|---|---|---|---|
| **XLM-RoBERTA** (`XLM-RoBERTa-Large`) | 1e-5 | 10 | 24 | 11 of 24 |
| **KB-BERT** (`bert-base-swedish-cased`) | 5e-5 | 10 | 24 | 10 of 12 |
| **mBERT** (`bert-base-multilingual-cased`) | 1e-4 | 5 | 64 | 11 of 12 |

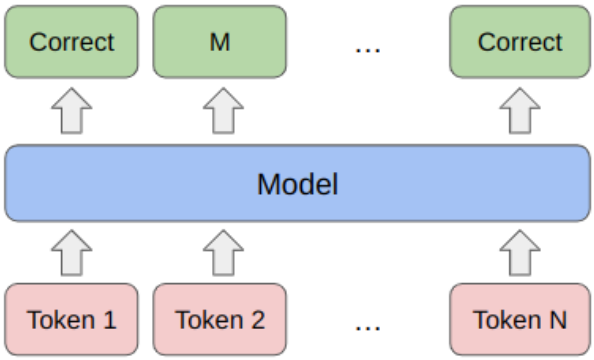


**Figure 5.3:** A diagram of multi token classification.

as `[Correct, B-M, I-M, ... , Correct]`. The `B-` signifies that the token marks the beginning of the error span, and `I-` signifies that the token is inside the error span. All other tokens are marked as correct.

We again fine-tuned mBERT, KB-BERT and XLM-RoBERTa for the task of multi-label token classification. More specifically, we trained one instance of each model to detect each of the five error types in DaLAJ-GED. For example, we trained a model to detect and label spans containing morphological errors, and another model to detect and label spans containing syntactical errors.

We froze the ten first layers of the models, and trained them with a learning rate of $10^{-5}$ for ten epochs with a batch size of 24.

## 5.2 Grammatical Error Correction

The experiments with GEC were conducted using both pre-trained encoder-decoder models, and pure decoder models, both fine-tuned on the task of GEC with the DaLAJ-GED dataset. We performed an additional experiment using the rule-based system Granska, and compared its results with the results of the fine-tuned transformer models. The pre-trained models were acquired from the Hugging Face platform.

### 5.2.1 Generating Synthetic Data

To generate synthetic data for the GEC task, we drew upon Table 2.6 which shows the most common errors in DaLAJ-GED. We created a proof-of-concept synthetic dataset using the most common errors. This synthetic data was used to extend DaLAJ-GED when fine-tuning mT5 models, described in the next section. We used the Swedish Wikipedia corpus as the basis for generating artificially corrupted sentences.

**en-ett errors.** Among the morphological errors, the articles *en* and *ett* are the most common confusions. Therefore, we extracted 10,000 sentences containing these words from the Swedish Wikipedia corpus and corrupted them by swapping *en* with *ett*, and vice versa, as well as dropping instances of *en* and *ett*.

**på-i errors.** Among the lexical errors, the prepositions *i* and *på* are the most common confusions. Therefore, we extracted 10,000 sentences containing these words from the Swedish Wikipedia corpus and corrupted them in the same fashion as described above.

**en-ett-på-i errors.** We additionally generated a synthetic dataset containing 10,000 sentences containing both the aforementioned *en-ett* errors and *på-i* errors.

**pronoun-verb order errors.** Casademont Moner and Volodina (2022) describe how they generated a small synthetic errorful dataset for Swedish. According to the authors, pronouns are the part-of-speech tag which produce the most verb order errors in the DaLAJ corpus. Therefore, we created artificial errors by swapping positions of pronouns and verbs in 10,000 sentences from the Swedish Wikipedia corpus.

**adverb-verb order errors.** In the same way, we generated 10,000 errorful sentences by swapping positions of adverbs and verbs in sentences from the Swedish Wikipedia corpus. This is because adverb-verb order errors are the second most common verb order errors according to Casademont Moner and Volodina (2022). For our proof-of-concept we only used a limited set of adverbs: *aldrig, alltid, sällan, ibland, ofta, ännu, fortfarande, redan, länge, genast, först, sist, då, nyss, förut, snart, inte, knappt, knappast, bara, nog, väl, ju, tyvärr, gärna, kanske, möjligen*.

### 5.2.2 Generating Correct Sentences

#### Granska

Granska is a program based on rule-based methods of GEC. We evaluated Granska on the test set of DaLAJ-GED to compare its performance with the transformer-based approaches. KTH Royal Institute of Technology provides Granska API [2] and a web page for Granska[3] where users can upload text files of input sentences. The Granska API returns a list of words without punctuations from the input sentence. In contrast, the web page for Granska returns a list

[2]https://skrutten.csc.kth.se/granskaapi/spell.php
[3]https://www.csc.kth.se/ viggo/stava

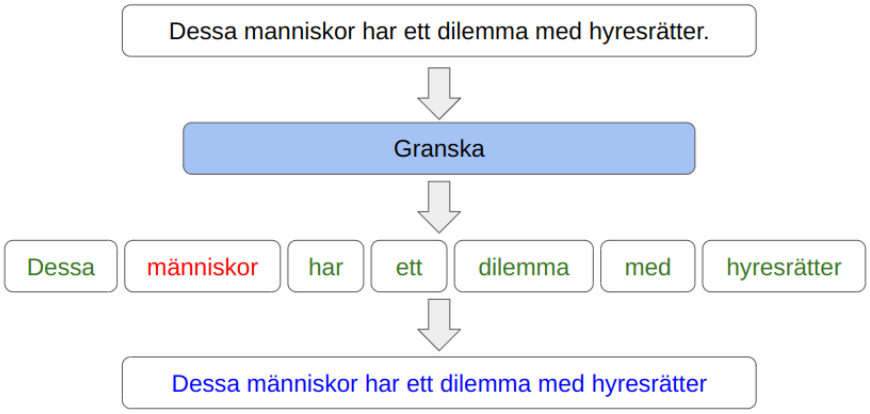


**Figure 5.4:** A diagram depicting how Granska corrects an input sentence and removes punctuation.

of the incorrect words and suggestions for each error. Therefore, We used the web page for Granska to obtain suggested corrections and replaced the incorrect words in the sentences. If Granska returns more than one suggestion for a word, we selected the first one. If the input sentences are already correct, the system should not propose any changes.

## mT5

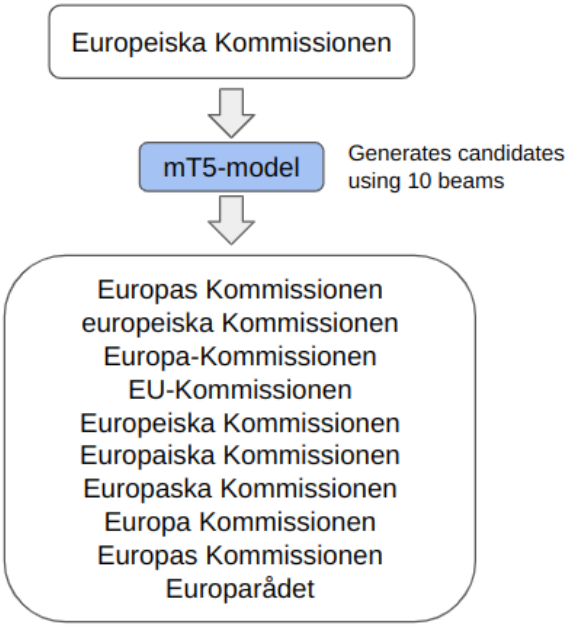


**Figure 5.5:** A depiction of how the mT5 model generates candidates with beam search using ten beams.

As described in Section 4, mT5 is a pre-trained sequence-to-sequence model that can be used for various tasks, such as machine translation. First, we fine-tuned an mT5 model on the task of generating correct sentences from incorrect ones.

It should be noted that "generating correct sentences" in this instance is a narrow definition meaning that the model's output precisely matches the corresponding gold sentence in the test set. For this reason, an output candidate sentence that is grammatically correct but does not exactly align with the gold sentence is considered incorrect.

After loading the mT5 model (`google/mt5-base`) from Hugging Face and fine-tuning it on DaLAJ-GED, we evaluated it on each sentence in the test set, by generating 10 output sentences with beam search, using 10 beams, as shown in Figure 5.5. Specifically, we probed how frequently the corresponding gold sentence could be found among the top 1, top 3, top 5 and top 10 output sentences.

In Section 3.2, we briefly described Rothe et al. (2021)'s work, in which mT5 was pretrained with a large synthetically corrupted corpus. Inspired by their work, we conducted a similar experiment to explore if synthetic errorful data can improve model performance. As described in Section 5.2.1, we generated five different sets of synthetic data. In total, we fine-tuned six mT5-models on the different datasets, the detail of which are shown in Table 5.3. The table also includes the designations by which we will refer to the various models from here on.

**Table 5.3:** The six mT5 models fine-tuned on DaLAJ-GED and synthetic data.

| Model designation | Fine-tuned on |
|---|---|
| **mT5** | DaLAJ-GED |
| **mT5-en-ett** | Synthetic Wikipedia data containing en-ett errors & DaLAJ-GED |
| **mT5-på-i** | Synthetic Wikipedia data containing på-i errors & DaLAJ-GED |
| **mT5-en-ett-på-i** | Synthetic Wikipedia data containing en-ett & på-i errors & DaLAJ-GED |
| **mT5-pron-verb** | Synthetic Wikipedia data containing pronoun-verb errors & DaLAJ-GED |
| **mT5-adv-verb** | Synthetic Wikipedia data containing adverb-verb errors & DaLAJ-GED |

According to the statistics in Section 2.3, approximately 99 % of the sentences in DaLAJ-GED have less than 50 tokens. Therefore, for these experiments and the following experiment with GPT-SW3, we used only sentences in the datasets which have fewer than 50 tokens, and set `max_token_length` to 128 for both the mT5 and GPT-SW3 models. We fine-tuned all mT5 models with a learning rate of $5 \cdot 10^{-4}$ for ten epochs with a batch size of 16.

## GPT-SW3

To fine-tune GPT-SW3 for GEC, the data was prepared in a specific way. For the training and validation data, the source and target sentences were combined with the separator `;`. The prefix `korr:` ("*korrigera*", *correct* in Swedish), was added before the source sentences. An example of the formatting of the input data for the training and validation sets is shown below:

```
korr: Jag flyga hem ikväll ; Jag flyger hem ikväll.
```

The model's input data during the prediction step was then formatted as follows:

```
korr: Jag bor i en hus.
```

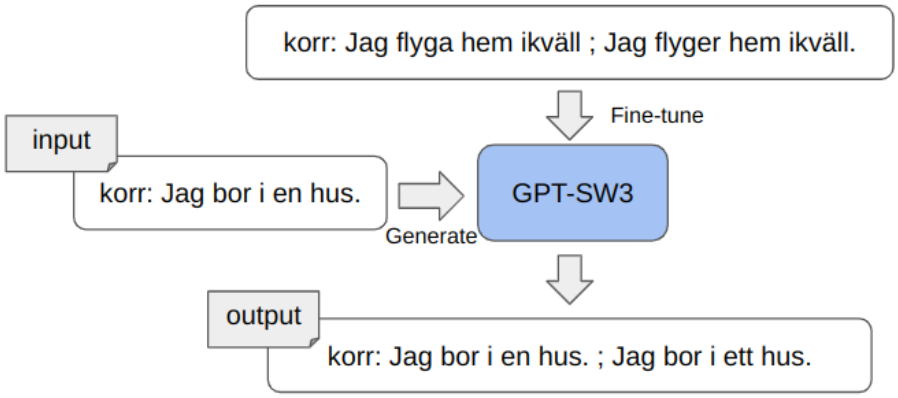


**Figure 5.6:** GPT-SW3 model generating output.

Figure 5.6 illustrates how GPT-SW3 is fine-tuned on the specially formatted training data. During the fine-tuning process, the model learns to generate text consisting of a prefix, an incorrect sentence, and a corrected sentence. Later, when an input sentence with a prefix and an incorrect sentence is provided during the prediction step, the model generates an output by appending the corrected sentence to the input.

Since we filtered out sentences containing more than 50 tokens, we expect the output sequences to contain at most approximately 100 tokens, and hence we set the `max_token_length` to 128. We used the GPT-SW3 model available on the Hugging Face platform (`AI-Sweden-Models/gpt-sw3-356m`), and fine-tuned it using a learning rate of $10^{-5}$ for three epochs with a batch size of 64 and a weight decay of 0.01. For the correction generation, we used beam search with ten beams and returned all ten output sentences.

## 5.3 Evaluation

The models trained on GED were evaluated with precision, recall and F1 score.

For the models trained on GEC, we compared the output sentences to the gold sentences and computed the accuracy of the top 1, 3, 5 and 10 generated output sentences. Furthermore, we computed the accuracy on each of the error types in DaLAJ-GED, using all 10 generated output sentences. In other words, we studied how often the correction to each error type could be found among the 10 output sentences.

The output sentences were also evaluated using the Generalized Language Evaluation Understanding (GLEU) metric, which was proposed by Napoles et al. (2015). If the gold sentence was found among the 10 output sentences, it was used. Otherwise the first output sentence was used. GLEU is a variant of the Bilingual Evaluation Understudy (BLEU) metric (Papineni et al., 2002), which is an evaluation method for translation tasks. In brief, a BLEU score is calculated as follows.

Supposing that a model generated two candidates, and there are two references, as shown below:
Candidate 1 : the the cat the the the cat.
Candidate 2 : The dog is on the mat.
Reference 1: The cat is on the mat.
Reference 2: The dogs are on the mat.

To calculate the BLEU score, we need to calculate the modified n-gram precision for each $n$.

$$\text{Modified n-gram precision}(p_n) = \frac{\sum_{n\text{-}gram \in Candidate} \text{Count}_{clip}(n\text{-}gram)}{\sum_{n\text{-}gram \in Candidate} \text{Count}(n\text{-}gram)}$$

$$\text{Count}_{clip} = \min(Count, Max\_Ref\_Count)$$

*Count* denotes the number of times which an n-gram appears in any of reference sentences, and *Max_Ref_Count* is the largest count observed in any single reference for that n-gram. In this example, if we calculate the bigram precision, the bigram *"the cat"* occurs twice in the Candidate 1, but only once in Reference 1. Therefore, $Count_{clip}$ of *"the cat"* is $\boldsymbol{min}(2, 1) = 1$. The total count of bigrams in Candidate 1 is 6, so therefore the modified bigram precision of Candidate 1 is $\frac{1}{6}$. Candidate 2 has 5 bigrams, and *"is on"*, *"on the"* and *"the mat"* appear at most once in Reference 1 and Reference 2. Therefore, the modified bigram precision of Candidate 2 is $\frac{3}{5}$.

$$\boldsymbol{BLEU} = \boldsymbol{BP} \times \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{5.1}$$

BLEU score is computed by Equation 5.1, where $\boldsymbol{N}$ is the max length of $\boldsymbol{n}$ and $\boldsymbol{w}$ is weight for different modified n-gram precision. For the baseline BLEU, $\boldsymbol{N}$ is set to 4 and $\frac{1}{4}$ is used for $\boldsymbol{w}$. $\boldsymbol{BP}$ is a brevity penalty for candidates that are shorter than the reference.

Napoles et al. (2015) noted that, specifically for the GEC task, words that do not change between the source and target sentences are likely correct, and should therefore not contribute as much to the scoring of the correction. GLEU is therefore a new metric well-suited to GEC, which gives more weight to the correctly changed words, and less weight to the words that should not be changed. The equation for GLEU identical to the one for BLEU, given by Equation 5.1. However, the equation for the modified precision is different, see Equation 5.2

$$p_n = \frac{\sum_{n\text{-}gram \in C} \text{Count}_{R\setminus S}(n\text{-}gram) - \lambda(\text{Count}_{S\setminus R}(n\text{-}gram)) + \text{Count}_{R}(n\text{-}gram)}{\sum_{n\text{-}gram \in C} \text{Count}_{R\setminus S}(n\text{-}gram) + \sum_{n\text{-}gram \in R\setminus S} \text{Count}_{R\setminus S}(n\text{-}gram)} \tag{5.2}$$

Where $\boldsymbol{C}$ denotes the candidate, $\boldsymbol{R}$ the reference and $\boldsymbol{S}$ the source sentence. The parameter $\lambda$ determines by how much incorrectly changed n-grams are penalized.

# Chapter 6
# Results

## 6.1 Grammatical Error Detection

### 6.1.1 Binary Sentence Classification

In this section, we present our results on the task of binary sentence classification described in Section 5.1.1.

#### Logistic Regression

**Table 6.1:** Baseline results of the logistic regression classifier using TF-IDF vectors and KB-BERT embeddings on the task of binary sentence classification.

| | | DaLAJ v1.0 | | | DaLAJ-GED | | |
|---|---|---|---|---|---|---|---|
| | **Class** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** |
| TF-IDF | Incorrect | 0.55 | 0.46 | 0.50 | 0.71 | 0.72 | 0.72 |
| | Correct | 0.54 | 0.63 | 0.58 | 0.73 | 0.72 | 0.72 |
| | Macro | 0.55 | 0.55 | 0.54 | 0.72 | 0.72 | 0.72 |
| KB-BERT embeddings | Incorrect | 0.61 | 0.63 | 0.62 | **0.81** | **0.81** | **0.81** |
| | Correct | 0.62 | 0.59 | 0.60 | **0.82** | **0.82** | **0.82** |
| | Macro | 0.62 | 0.61 | 0.61 | **0.81** | **0.81** | **0.81** |

Table 6.1 shows the results of logistic regression classifier with TF-IDF vectors and KB-BERT embeddings. When comparing the results of the TF-IDF vectors with those of the KB-BERT embeddings, we can see that using KB-BERT embeddings yields higher F1-scores than when using TF-IDF vectors, for both datasets. Additionally, when comparing the results of

the datasets to each other, we can see that the models trained on DaLAJ-GED perform better than those trained on DaLAJ v1.0. Therefore, the model with KB-BERT embeddings trained on DaLAJ-GED has the best F1-score, which is 0.81.

### Fine-tuned Encoder Models

**Table 6.2:** Results for the encoder models fine-tuned using the DaLAJ datasets on the task of binary sentence classification.

| | | DaLAJ v1.0 | | | DaLAJ-GED | | |
|---|---|---|---|---|---|---|---|
| | **Class** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** |
| BERT | Incorrect | 0.52 | 0.42 | 0.46 | 0.70 | 0.73 | 0.72 |
| | Correct | 0.51 | 0.61 | 0.56 | 0.73 | 0.70 | 0.72 |
| | Macro | 0.52 | 0.51 | 0.51 | 0.71 | 0.72 | 0.72 |
| RoBERTa | Incorrect | 0.52 | 0.63 | 0.57 | 0.68 | **0.80** | 0.73 |
| | Correct | 0.52 | 0.41 | 0.46 | 0.77 | 0.64 | 0.69 |
| | Macro | 0.52 | 0.52 | 0.52 | 0.72 | 0.72 | 0.71 |
| mBERT | Incorrect | 0.67 | 0.19 | 0.30 | 0.77 | 0.75 | 0.76 |
| | Correct | 0.53 | 0.91 | 0.67 | 0.76 | 0.78 | 0.77 |
| | Macro | 0.60 | 0.55 | 0.48 | 0.76 | 0.76 | 0.76 |
| XLM-RoBERTa | Incorrect | 0.73 | 0.30 | 0.43 | 0.80 | 0.79 | 0.79 |
| | Correct | 0.56 | 0.89 | 0.69 | **0.80** | 0.80 | 0.80 |
| | Macro | 0.65 | 0.60 | 0.56 | 0.80 | 0.80 | 0.80 |
| KB-BERT | Incorrect | 0.71 | 0.66 | 0.68 | **0.92** | 0.73 | **0.82** |
| | Correct | 0.68 | 0.73 | 0.70 | 0.78 | **0.94** | **0.85** |
| | Macro | 0.69 | 0.69 | 0.69 | **0.85** | **0.84** | **0.83** |

Table 6.2 presents the results of the encoder models that were directly fine-tuned for binary sentence classification. First, comparing the results from DaLAJ v1.0 and DaLAJ-GED datasets, we observe that all models performed better when trained on DaLAJ-GED. It is worth noting that the F1-scores on DaLAJ v1.0 are all lower than 0.70, whereas those on DaLAJ-GED are all higher than 0.70.

On DaLAJ v1.0, KB-BERT yields the highest F1 score of 0.69, followed by XLM-RoBERTa and RoBERTa. KB-BERT performs the best on DaLAJ-GED as well, with an F1 score of 0.83, followed by XLM-RoBERTa and mBERT.

## 6.1.2 Binary Token Classification

We present our results for the task of binary token classification described in Section 5.1.2.

Table 6.3 displays the performances of XLM-RoBERTa, mBERT and KB-BERT on the task of binary token classification. The result shows that XLM-RoBERTa performs the best, having a macro-averaged F1 score of 0.71, with KB-BERT following it with a macro F1 score of 0.66.

**Table 6.3:** Results for the encoder models fine-tuned using the DaLAJ-GED dataset on the task of binary token classification.

| | XLM-RoBERTa | | | mBERT | | | KB-BERT | | |
|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** |
| Incorrect | 0.68 | **0.61** | **0.64** | 0.62 | 0.13 | 0.21 | **0.73** | 0.49 | 0.59 |
| Correct | **0.79** | **0.78** | **0.78** | 0.71 | 0.59 | 0.65 | 0.74 | 0.70 | 0.72 |
| Macro | **0.74** | **0.69** | **0.71** | 0.67 | 0.36 | 0.43 | 0.73 | 0.60 | 0.66 |

### 6.1.3 Multi-label Token Classification

We present our results for the task of multi-label token classification, using the models described in Section 5.1.3.

**Table 6.4:** Results for the encoder models fine-tuned using the DaLAJ-GED dataset on the task of multi-label token classification.

| | XLM-RoBERTa | | | mBERT | | | KB-BERT | | |
|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** | **Precision** | **Recall** | **$F_1$** |
| P | 0.43 | **0.66** | 0.52 | **0.88** | 0.33 | 0.48 | 0.78 | 0.41 | **0.54** |
| O | **0.74** | **0.69** | **0.71** | 0.63 | 0.25 | 0.36 | 0.73 | 0.45 | 0.56 |
| L | 0.48 | **0.32** | **0.38** | 0.11 | 0.00 | 0.00 | **0.63** | 0.06 | 0.11 |
| M | **0.71** | **0.74** | **0.73** | 0.61 | 0.11 | 0.19 | 0.66 | 0.52 | 0.58 |
| S | 0.50 | **0.48** | **0.49** | **0.64** | 0.05 | 0.10 | 0.55 | 0.25 | 0.34 |
| Macro | 0.57 | **0.58** | **0.57** | 0.57 | 0.15 | 0.23 | **0.67** | 0.34 | 0.50 |

Table 6.4 shows the performance of XLM-RoBERTa, mBERT and KB-BERT on multi-label token classification task. Compared to the other models, XLM-RoBERTa model achieves better recall and F1 scores on almost all classes, while KB-BERT has the highest precision. Regarding precision, mBERT shows the highest precision for P and S classes among the three models, whereas KB-BERT has the highest for L class.

## 6.2 Grammatical Error Correction

In this section, we present our results on GEC by Granska and the encoder-decoder and decoder models described in Section 5.2.2. We investigate not only how well the models correct the incorrect sentences, but also how good they are at copying correct input sentences to the output, without introducing any changes.

### 6.2.1 Granska

As shown in Table 6.5, it achieves a 95% accuracy on the correct sentences, meaning it falsely interprets 5% of them as being incorrect, and suggests alterations for them. The accuracy

**Table 6.5:** The accuracy of Granska's GEC on the DaLAJ-GED test set.

| Sentence Type | Nbr. examples in test data | Accuracy |
|---|---|---|
| **Correct** | 2,214 | 0.95 |
| **Incorrect** | 2,095 | 0.07 |
| **Whole test set** | 4,309 | 0.52 |

on incorrect sentences is 7%, meaning that Granska only produces the correct gold sentence from 7% of the incorrect sentences.

**Table 6.6:** The accuracy of Granska's GEC, grouped by DALAJ-GED's error types.

| Error Type | Nbr. examples in test data | Accuracy |
|---|---|---|
| **M** | 646 | 0.01 |
| **S** | 424 | 0.22 |
| **L** | 357 | 0 |
| **O** | 343 | 0.12 |
| **P** | 171 | 0 |
| **O+S** | 6 | 0.33 |

We studied the error types of the incorrect sentences more closely. The results are displayed in Table 6.6, and we can observe that Granska corrects S and O-type errors slightly better than M-, L- and P-type errors.

The results were also evaluated by the GLEU metric. The GLEU score on whole test set was 0.87, and 0.77 on the incorrect sentences in the test set.

### 6.2.2 Encoder-decoder Models

We conducted experiments on GEC using the encoder-decoder model mT5 as described in Section 5.2.2. Table 6.7 illustrates how frequently the correct sentence can be found among the top 1, top 3, top 5 and top 10 output sentences that were generated by the mT5 models.

We observed that the baseline mT5 model trained solely on DaLAJ-GED, manages to generate a perfectly matched sentence 70% of the time among its top 10 outputs. However, the most probable output sentence (i.e. the top 1) is only correct 46% of the time. Most of the mT5 models trained on both synthetic data and DaLAJ-GED perform better for this task. The mT5-på-i model generated the correct sentence 76% of the time among its 10 outputs, which is higher than the performance of the baseline mT5 model. Similarly, the mT5-en-ett-på-i model generated the gold sentence 55% of the time with its top 1 outputs, which is nearly 10% more frequently than the baseline mT5 model. Table 6.8 shows the GLEU scores for the encoder-decoder mT5 models.

We further studied if the mT5-en-ett model performed better on sentences containing morphological errors involving *en* or *ett* compared to the baseline mT5. The corresponding

**Table 6.7:** The accuracy, i.e. how frequently the mT5 models manage to generate an output sentence that matches the gold sentence among their top 1, 3, 5 and 10 generated outputs. The whole test set, including both correct and incorrect sentences, was used for this experiment.

| | Accuracy | | | |
|---|---|---|---|---|
| | **1 candidate** | **3 candidates** | **5 candidates** | **10 candidates** |
| **mT5** | 0.46 | 0.61 | 0.65 | 0.70 |
| **mT5-en-ett** | 0.52 | 0.66 | 0.70 | 0.74 |
| **mT5-på-i** | 0.54 | **0.68** | **0.73** | **0.76** |
| **mT5-en-ett-på-i** | **0.55** | **0.68** | 0.71 | 0.75 |
| **mT5-pron-verb** | 0.51 | 0.62 | 0.66 | 0.70 |
| **mT5-adv-verb** | 0.40 | 0.54 | 0.58 | 0.63 |

**Table 6.8:** GLEU scores for the mT5 models.

| | **Whole test set** | **Incorrect sentences** |
|---|---|---|
| **mT5** | 0.91 | 0.87 |
| **mT5-en-ett** | 0.92 | 0.88 |
| **mT5-på-i** | **0.93** | **0.89** |
| **mT5-en-ett-på-i** | 0.92 | 0.88 |
| **mT5-pron-verb** | 0.89 | 0.86 |
| **mT5-adv-verb** | 0.91 | 0.86 |

**Table 6.9:** The hit rate of mT5 and mT5-en-ett on sentences containing specific errors.

| | | Accuracy | |
|---|---|---|---|
| **Error type** | **Nbr. examples in test data** | **mT5** | **mT5-en-ett** |
| **M-errors containing en-ett** | 84 | 0.58 | **0.67** |
| **Other incorrect sentences** | 2011 | 0.52 | **0.54** |
| **Correct sentences** | 2214 | 0.87 | **0.93** |

results are shown in Table 6.9. The baseline mT5 model has an accuracy of 58% among its top 10 output sentences, whereas the mT5-en-ett performs better, having a 67% accuracy on the morphological errors involving *en* or *ett*. Furthermore, the accuracy on the rest of the test set also improved for the mT5-en-ett model.

In the same way, we also studied whether the mT5-på-i model improved the performance on sentences containing lexical and syntactical errors involving *på* or *i*. Table 6.10 shows that the baseline mT5 entirely fails to correct these sentences in the test set, but that the model trained on the synthetic data (mT5-på-i) corrects the sentences containing lexical errors with 68% accuracy, and the sentences containing syntactical errors with 57% accuracy. Furthermore, the mT5-på-i model performs as well as the baseline mT5 model on the rest of the test set's incorrect sentences, and is slightly better at reconstructing the test set's correct sentences.

**Table 6.10:** The accuracy of mT5 and mT5-på-i on sentences containing specific errors.

| Error type | Nbr. examples in test data | Accuracy | |
|---|---|---|---|
| | | mT5 | mT5-på-i |
| **L-errors containing på-i** | 19 | 0 | **0.68** |
| **S-errors containing på-i** | 21 | 0 | **0**.57 |
| **Other incorrect sentences** | 2055 | **0.52** | **0.52** |
| **Correct sentences** | 2214 | 0.87 | **0.95** |

We further investigated the mT5 models' accuracy grouped by DaLAJ-GED's error types. In Table 6.11, we observe that some error types are easier than others to correct. Morphological and orthographic errors are easiest to correct, followed by syntactical errors. Lexical errors are more difficult, and most difficult are punctuation errors. Even though sentences in DaLAJ-GED only contain one error each, the error may cover multiple error tag types. These examples are not as common in the dataset, but the model still performs better for some of these combinations (e.g. morphological + orthographic) than for the more difficult singular error types.

**Table 6.11:** The encoder-decoder models' accuracy, grouped by DaLAJ-GED's error types. Some sentences contain multiple error types. The top 10 output sentences generated by the models were considered in this experiment.

| Error Type | Nbr. examples in test data | Accuracy | | | | | |
|---|---|---|---|---|---|---|---|
| | | mT5 | mT5-en-ett | mT5-på-i | mT5-en-ett-på-i | mT5-pron-verb | mT5-adv-verb |
| **M** | 646 | 0.67 | 0.72 | **0.75** | 0.71 | 0.54 | 0.65 |
| **S** | 424 | 0.46 | 0.49 | **0.51** | 0.47 | 0.36 | 0.48 |
| **L** | 357 | 0.32 | 0.35 | **0.39** | 0.33 | 0.27 | 0.29 |
| **O** | 343 | 0.71 | 0.71 | **0.73** | 0.68 | 0.62 | 0.66 |
| **P** | 171 | 0.19 | 0.20 | **0.22** | 0.19 | 0.19 | 0.19 |
| **M+O** | 57 | 0.44 | 0.53 | **0.56** | 0.53 | 0.32 | 0.49 |
| **L+S** | 48 | **0.48** | **0.48** | 0.46 | 0.44 | 0.29 | 0.44 |
| **L+M** | 21 | 0.19 | 0.14 | 0.24 | 0.19 | 0.10 | **0.33** |
| **M+S** | 11 | **0.36** | 0.18 | 0.18 | 0.09 | 0.19 | 0.09 |
| **L+O** | 6 | 0.17 | 0.17 | **0.33** | 0.17 | 0.17 | 0.17 |
| **O+S** | 6 | 0.33 | **0.50** | 0.33 | 0.33 | 0 | 0.33 |
| **L+M+O** | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **M+O+S** | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **L+O+S** | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### 6.2.3 Decoder Models

We conducted identical experiments using the decoder-based GPT-SW3 model described in Section 5.2.2. Table 6.7 illustrates how frequently the correct sentence can be found among the top 1, top 3, top 5 and top 10 output sentences that were generated by the GPT-SW3 model.

**Table 6.12:** The accuracy, i.e. how frequently the GPT-SW3 model manages to generate an output sentence that matches the gold sentence among their top 1, 3, 5 and 10 generated outputs. The whole test set, including both correct and incorrect sentences, was used for this experiment.

| | Accuracy | | | |
|---|---|---|---|---|
| | **1 candidate** | **3 candidates** | **5 candidates** | **10 candidates** |
| **GPT-SW3** | 0.57 | 0.71 | 0.76 | 0.80 |

We observed that the GPT-SW3 model, which was trained solely on DaLAJ-GED, manages to generate a perfectly matched sentence 80% of the time among its top 10 outputs, and that the most probable output sentence (i.e. the top 1) is correct 57% of the time. Its GLEU score was 0.93 on the whole test set, and 0.90 on the incorrect sentences of the test set.

**Table 6.13:** The decoder-based GPT-SW3's accuracy, grouped by DaLAJ-GED's error types. The top 10 output sentences generated by GPT-SW3 were considered in this experiment.

| **Error Type** | **Nbr. examples in test data** | Accuracy **GPT-SW3** |
|---|---|---|
| **M** | 646 | 0.77 |
| **S** | 424 | 0.56 |
| **L** | 357 | 0.48 |
| **O** | 343 | 0.78 |
| **P** | 171 | 0.47 |
| **M+O** | 57 | 0.60 |
| **L+S** | 48 | 0.58 |
| **L+M** | 21 | 0.29 |
| **M+S** | 11 | 0.27 |
| **L+O** | 6 | 0.17 |
| **O+S** | 6 | 0.50 |
| **L+M+O** | 2 | 0 |
| **M+O+S** | 1 | 0 |
| **L+O+S** | 1 | 0 |

Analogous to the experiments with mT5, we investigated GPT-SW3's accuracy grouped by the DaLAJ-GED's error types. The results are shown in Table 6.13.

### 6.2.4 Comparative Analysis

We compare the results of the baseline mT5 model, the best mT5 models', and those of GPT-SW3. In Table 6.14 we see that GPT-SW3 outperforms the best mT5 models on the task of generating the correct gold sentence, both as its most probable output, and among its top 10 beam search candidates.

**Table 6.14:** The table shows a comparison between the baseline mT5 model, the best mT5 models, and GPT-SW3, on how frequently they generate an output sentence that matches the gold sentence among their top 1, 3, 5 and 10 outputs.

| | Accuracy | | | |
|---|---|---|---|---|
| | **1 candidate** | **3 candidates** | **5 candidates** | **10 candidates** |
| **Baseline mT5 model** | 0.46 | 0.61 | 0.65 | 0.70 |
| **Best mT5 model** | 0.55 | 0.68 | 0.73 | 0.76 |
| **GPT-SW3** | **0.57** | **0.71** | **0.76** | **0.80** |

Moreover, in Table 6.15 we see that GPT-SW3 performs better at correcting most error types, with the exception of `L+M`, `M+S` and `L+O`. Finally, Table 6.16 shows a comparison of the GLEU scores of the baseline mT5 model, the best mT5 model and GPT-SW3.

**Table 6.15:** A comparison of the accuracy of the baseline mT5 trained only on DaLAJ-GED, GPT-SW3, also trained only on DaLAJ-GED, and the best mT5 models (grouped by DaLAJ-GED's error types).

| | | Accuracy | | |
|---|---|---|---|---|
| **Error Type** | **Nbr. examples in test data** | **Baseline mT5** | **Best mT5 model** | **GPT-SW3** |
| **M** | 646 | 0.67 | 0.75 | **0.77** |
| **S** | 424 | 0.46 | 0.51 | **0.56** |
| **L** | 357 | 0.32 | 0.39 | **0.48** |
| **O** | 343 | 0.71 | 0.73 | **0.78** |
| **P** | 171 | 0.19 | 0.22 | **0.47** |
| **M+O** | 57 | 0.44 | 0.56 | **0.60** |
| **L+S** | 48 | 0.48 | 0.48 | **0.58** |
| **L+M** | 21 | 0.19 | **0.33** | 0.29 |
| **M+S** | 11 | **0.36** | **0.36** | 0.27 |
| **L+O** | 6 | 0.17 | **0.33** | 0.17 |
| **O+S** | 6 | 0.33 | **0.50** | **0.50** |
| **L+M+O** | 2 | 0 | 0 | 0 |
| **M+O+S** | 1 | 0 | 0 | 0 |
| **L+O+S** | 1 | 0 | 0 | 0 |

**Table 6.16:** Comparison between GLEU scores of the baseline mT5 model, the best mT5 model and GPT-SW3.

| | Whole test set | Incorrect sentences |
|---|---|---|
| **Baseline mT5 model** | 0.91 | 0.87 |
| **Best mT5 model** | **0.93** | 0.89 |
| **GPT-SW3** | **0.93** | **0.90** |

# Chapter 7
# Discussion

In this chapter, we discuss the results from our experiments and propose directions for future work.

## 7.1 Interpretations of Results

In our experiments with binary sentence classification we trained models on two different datasets, DaLAJ v1.0 and DaLAJ-GED. Since DaLAJ-GED is approximately five times larger than DaLAJ v1.0, we anticipated that models trained on DaLAJ-GED would perform better than those on DaLAJ v1.0. The experimental results on binary sentence classification indicate that this hypothesis was correct.

We also performed an experiment using TF-IDF vectors and KB-BERT embeddings as input features for training a logistic regression classifier, and compared the results. The classifier trained using KB-BERT embeddings performed better than the one trained on TF-IDF vectors. The context of each word in a sentence is important for GED tasks, and we conclude that KB-BERT embeddings are better at encapsulating this context and therefore improve the model's performance.

In addition, we also directly fine-tuned encoder models for the task of binary sentence classification by training a small neural network classifier on top of the encoders. The multilingual models such as mBERT and XLM-RoBERTa produced better results than their monolingual English counterparts, BERT and RoBERTa. Interestingly, KB-BERT, which is a monolingual Swedish model, performed even better than the multilingual models. This suggests that the use of monolingual models may be beneficial for GED.

In our experiments on token classification, both binary and multi-label token classification, we noted that XLM-RoBERTa showed better results than mBERT and KB-BERT. This was expected because we used XLM-RoBERTa-Large, and it is the largest among the models, with 580 million parameters compared to the BERT models' 110 million parameters. We conclude that the size of the model can have a significant impact on its performance. However,

KB-BERT exhibited a better precision than XLM-RoBERTa for multi-label token classification despite being smaller. Precision is crucial when providing feedback to second language learners, as imprinting incorrect information on the learner can be detrimental and lead to confusion. XLM-RoBERTa has a high precision on O and M-type errors, but the precision of P and L-type errors are below 0.5. On the other hand, KB-BERT yielded a precision higher than 0.5 for all classes.

For GEC, we utilized encoder-decoder and decoder-based models to produce a correct output sentence from input sentences. To comparatively evaluate the models' performances, we first studied the performance of the rule-based model, Granska. We expected that Granska would produce good results, at least for orthographic errors, which are related to spelling errors. However, it only corrected 7% of the incorrect sentences and 12% of sentences containing orthographic errors.

The transformer-based models performed better. The baseline mT5 model fine-tuned only on DaLAJ-GED generated a correct output sentences for 70% of the test set when beam search with 10 beams and 10 return sequences was used. However, when using only one beam, and producing one output sentence, it was correct for only 46% of the test set. In other words, the sentence that the model considers to be the most probable one, is only correct 46% of the time. GPT-SW3, when trained on DaLAJ-GED, performed even better, achieving an 80% hit rate among its 10 beam search candidates, and a 57% hit rate among its most probable output sentences. From this observation, we conclude that the transformer-models are largely capable of generating corrected sentences, but that the models' internal language models are yet insufficient for the task of ranking the correct sentences as the most probable ones.

We also observed that synthetic data improves GEC models' performance. The models mT5-en-ett, mT5-på-i and mT5-en-ett-på-i models outperformed the baseline mT5 model. These models not only showed improved performance on the specific errors that they were fine-tuned on, but also enhanced the performance overall. This demonstrates the potential of using synthetic data for improving GEC models. However, mT5-pron-verb did not show significant improvements, and the results of mT5-adv-verb were worse than the baseline mT5 model. We speculate that this is because we generated 10,000 sentences using a small list of only 27 adverbs, which likely introduced a heavy bias to the training data, which lowered overall performance. We believe that the performance would be better if the synthetic data for adverb-verb errors was generated using a large lexicon of adverbs.

The GLEU scores of the models are quite high, but we conclude that this is due to the format of the DaLAJ dataset, as each sentence in DaLAJ only contains a single error. If each sentence contained multiple errors, there would be more work to do for the models, and potentially more mistakes to make, which could yield lower GLEU scores.

## 7.2 Future work

The hyperparameters of a model, such as its learning rate, batch size, weight decay, warm-up and number of epochs, can play a key role in determining the model's performance. Due to time and compute limitations, we were unable to thoroughly explore multiple options for each hyperparameter. Ideally, one would perform a grid search to find the best parameters by optimizing the performance on the validation data.

While we discussed the potential of using synthetic data to improve GEC models, we did not use synthetic data for the experiments involving GED. It would be interesting to also train GED models on synthetic data and evaluate its effect on their performance. Additionally, the synthetic data can be generated in different ways, including adding more types of errors and using more or less synthetic data. It is also possible to use more generic types of synthetic errors, as Rothe et al. (2021) did, as opposed to ones involving specific words or parts of speech, as we did. Lastly, we propose the use of Stahlberg and Kumar (2021)s method of generating synthetic data whose distribution of errors matches that of the authentic corpus.

Since GPT-SW3 generally performed better than the mT5 models, despite being trained only on DaLAJ-GED, it would be especially interesting to attempt to fine-tune GPT-SW3 on synthetic data as well.

While studying mT5 and GPT-SW3 and their generation of different candidates corrections, we realized the importance of having a system that proposes the correct sentence as the top candidate. We investigated methods that could solve this problem and we found that using a re-ranker is a promising alternative. The re-ranker would be trained to learn the mappings between a source sentence and a list of potential candidate corrections. In a live deployment scenario, the re-ranker would score the generated candidates, and its score would be multiplied with the encoder-decoder or decoder-based model's own score, to produce a new ranking of the candidates. Options for re-rankers include Sentence BERT (Reimers and Gurevych, 2019) and Transcormer (Song et al., 2022).

Although encoder-decoder and decoder-based models like mT5 and GPT-SW3 have shown promising results in GEC tasks, they still have limitations. For one, the decoders can only consider the left-hand context when evaluating each token in a sentence, which means that valuable contextual data is lost. The models can also hallucinate unwanted outputs, so using a pure encoder model may yield more interpretable results than an encoder-decoder or decoder-based model. Additionally, systems that perform GEC by sequence-tagging instead of text generation, such as GECToR (Omelianchuk et al., 2020), result in much faster data processing. A system like GECToR is largely trained on synthetic data and could be trained for the Swedish language.

While the field of NLP has moved in the direction of auto-regressive models like GPT, we still see a great value in encoder-based models for GEC due to the reasons mentioned above. Furthermore, the results in Section 6.2 show that for a low-resource language like Swedish, the multilingual sequence-to-sequence model mT5 performed fairly well compared to a similarly sized GPT-model that was trained only on Swedish data. We believe that the sequence-to-sequence nature of encoder-decoder models makes them well suited to the task of NMT, and including GEC.

## 7.3 Conclusion

In this thesis, we investigated the resources available for GED and GEC. Swedish annotated corpora for GED and GEC are sparse. Examples such as SweLL and DaLAJ, have been presented only in recent years, and the new dataset, DaLAJ-GED, was released in 2023. We compared different models trained on DaLAJ v1.0 and DaLAJ-GED and observed that the larger DaLAJ-GED dataset yielded better model performance.

The rule-based model Granska was one of the early models designed for GEC in Swedish,

developed in the early 2000s. Since the introduction of BERT in 2018, several Swedish version of BERT have been developed and KB-BERT is one of them. In 2023, a generative decoder model trained largely on Swedish data, GPT-SW3, was released. Comparisons of KB-BERT's performance on GED tasks with multilingual BERT models, such as mBERT and XLM-RoBERTa, show that KB-BERT performs better on sentence classification and produced the best precision for multi-label token classification even when compared with the much larger XLM-RoBERTa-Large. We also used GPT-SW3 to perform GEC and found that it outperformed the multilingual mT5 model. This suggests that well-trained monolingual models may perform well for GED and GEC.

We were interested in the possibility of combining old techniques such as rule-based methods with new transformer-based approaches in order to improve the performance of GED and GEC. However, we found that the rule-based system Granska was only able to correct 7% of the incorrect sentences, and we concluded that it would not improve the performance of other models even if used in an ensemble.

We specifically evaluated the models on each error type in DaLAJ-GED, in order to see which are more difficult to detect and correct. According to Table 6.4 and Table 6.11, the lexical and syntactical errors are more difficult to detect and correct than the morphological and orthographic. In an attempt to alleviate this issue, we generated synthetic data containing some specific lexical and syntactical errors and observed that it did improve the model's performance for these error types. We propose the use of synthetic data for training the Swedish models KB-BERT and GPT-SW3, to enhance their performance of GED and GEC in Swedish.

While the field of GED and GEC in Swedish remains relatively underexplored and challenging, the application of innovative techniques such as machine learning show great promise. With further research and development, we anticipate more effective and accurate models shortly, that will likely greatly benefit both language learners and native users of the Swedish language.

# References


Arppe, A. (2000). Developing a grammar checker for Swedish. In *Proceedings of the 12th Nordic Conference of Computational Linguistics (NODALIDA 1999)*, pages 13–27, Trondheim, Norway.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs.CL].

Bigert, J. (2002). Robust Error Detection: A Hybrid Approach Combining Unsupervised Error Detection and Linguistic Knowledge. In *Proceedings of 2nd Workshop Robust Methods in Analysis of Natural language Data (ROMAND'02)*, pages 10–19, Frascati, Italy.

Birn, J. (2000). Detecting grammar errors with Lingsoft's Swedish grammar checker. In *Proceedings of the 12th Nordic Conference of Computational Linguistics (NODALIDA 1999)*, pages 28–40, Trondheim, Norway. Department of Linguistics, Norwegian University of Science and Technology, Norway.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. Zenodo.

Bryant, C., Felice, M., and Briscoe, T. (2017). Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada.

Carlberger, J., Domeij, R., Kann, V., and Knutsson, O. (2004). The development and performance of a grammar checker for Swedish : A language engineering perspective. *Natural Language Engineering*, 1.

Casademont Moner, J. and Volodina, E. (2022). Generation of Synthetic Error Data of Verb Order Errors for Swedish. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 33–38, Seattle, Washington, USA.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259 [cs.CL].

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs.CL].

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2019). Unsupervised Cross-lingual Representation Learning at Scale. arXiv:1911.02116 [cs.CL].

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL].

Ekgren, A., Cuba Gyllensten, A., Gogoulou, E., Heiman, A., Verlinden, S., Öhman, J., Carlsson, F., and Sahlgren, M. (2022). Lessons Learned from GPT-SW3: Building the First Large-Scale Generative Language Model for Swedish. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3509–3518, Marseille, France.

Felice, M., Bryant, C., and Briscoe, T. (2016). Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. arXiv:1705.03122 [cs.CL].

Google Research (2019). multilingual.md. `https://github.com/google-research/bert/blob/master/multilingual.md`. GitHub ReadMe file.

Heidorn, G. E., Jensen, K., Miller, L. A., Byrd, R. J., and Chodorow, M. S. (1982). The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S., and Heafield, K. (2018). Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana, USA.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2017). Neural Machine Translation in Linear Time. arXiv:1610.10099 [cs.CL].

Kaneko, M. and Komachi, M. (2019). Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection. arXiv:1904.07334 [cs.CL].

Kann, V., Domeij, R., Hollman, J., and Tillenius, M. (1998). Implementation Aspects and Applications of a Spelling Correction Algorithm. *Journal of Quantitative Linguistics*.

Knutsson, O. (2005). *Developing and Evaluating Language Tools for Writers and Learners of Swedish*. PhD thesis, KTH Royal Institute of Technology.

KTH Royal Institute of Technology (2020). Probgranska. https://skrutten.csc.kth.se/granskaapi/probcheck. Accessed: 2023-05-06.

Lample, G. and Conneau, A. (2019). Cross-lingual Language Model Pretraining. arXiv:1901.07291 [cs.CL].

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:1907.11692 [cs.CL].

Malmsten, M., Börjeson, L., and Haffenden, C. (2020). Playing with Words at the National Library of Sweden – Making a Swedish BERT. arXiv:2007.01658 [cs.CL].

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL].

Miller, L. (1980). Project EPISTLE: A System for the Automatic Analysis of Business Correspondence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 280–282.

Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. (2015). Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China.

Norlund, T. and Stenbom, A. (2021). Building a Swedish Open-Domain Conversational Language Model. In *Nordic Conference of Computational Linguistics*.

Omelianchuk, K., Atrasevych, V., Chernodub, A., and Skurzhanskyi, O. (2020). GECToR–Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Online.

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.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. Preprint. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG].

Rei, M. and Søgaard, A. (2018). Jointly Learning to Label Sentences and Tokens. arXiv:1910.10683 [cs.CL].

Rei, M. and Yannakoudakis, H. (2016). Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084 [cs.CL].

Rothe, S., Mallinson, J., Malmi, E., Krause, S., and Severyn, A. (2021). A Simple Recipe for Multilingual Grammatical Error Correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online.

Salazar, J., Liang, D., Nguyen, T. Q., and Kirchhoff, K. (2020). Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online.

Schuster, M. and Nakajima, K. (2012). Japanese and Korean Voice Search. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152.

Sjöbergh, J. and Knutsson, O. (2005). Faking Errors to Avoid Making Errors: Very Weakly Supervised Learning for Error Detection in Writing. In *Proceedings of RANLP 2005*, page 506–512, Borovets, Bulgaria.

Song, K., Leng, Y., Tan, X., Zou, Y., Qin, T., and Li, D. (2022). Transcormer: Transformer for Sentence Scoring with Sliding Language Modeling. arXiv:2205.12986 [cs.CL].

Stahlberg, F. and Kumar, S. (2021). Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online.

Stymne, S. and Ahrenberg, L. (2010). Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 2175–2181, Valletta, Malta.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215 [cs.CL].

Swedish Public Employment Service (2020). Swedish BERT models. `https://github.com/af-ai-center/SweBERT`. GitHub Repository.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All You Need. arXiv:1706.03762 [cs.CL].

Volodina, E., Granstedt, L., Matsson, A., Megyesi, B., Pilán, I., Prentice, J., Rosén, D., Rudebeck, L., Schenström, C.-J., Sundberg, G., and Wirén, M. (2019). The SweLL Language Learner Corpus. In *Northern European Journal of Language Technology*, volume 6, pages 67–104. Linkoping University Electronic Press.

Volodina, E., Mohammed, Y. A., Berdicevskis, A., Bouma, G., and Öhman, J. (2023). DaLAJ-GED - a dataset for Grammatical Error Detection tasks on Swedish. In *Proceedings of the 12th Workshop on Natural Language Processing for Computer-Assisted Language Learning*, pages 94–101, Tórshavn, Faroe Islands.

Volodina, E., Mohammed, Y. A., and Klezl, J. (2021). DaLAJ - a dataset for linguistic acceptability judgments for Swedish: Format, baseline, sharing. arXiv:2105.06681 [cs.CL].

Volodina, E., Pilán, I., Rødven Eide, S., and Heidarsson, H. (2014). You Get what You Annotate: A Pedagogically Annotated Corpus of Coursebooks for Swedish as a Second Language. In *Proceedings of the third workshop on NLP for computer-assisted language learning*, pages 128–144, Uppsala, Sweden.

Weng, Y., Miryala, S. S., Khatri, C., Wang, R., Zheng, H., Molino, P., Namazifar, M., Papangelis, A., Williams, H., Bell, F., and Tur, G. (2020). Joint Contextual Modeling for ASR Correction and Language Understanding. arXiv:2002.00750 [cs.CL].

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online.

Yasunaga, M., Leskovec, J., and Liang, P. (2021). LM-Critic: Language Models for Unsupervised Grammatical Error Correction. arXiv:2109.06822 [cs.CL].

Yuan, Z., Taslimipoor, S., Davis, C., and Bryant, C. (2021). Multi-Class Grammatical Error Detection for Correction: A Tale of Two Systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-02

**EXAMENSARBETE** Correction of grammatical errors in Swedish
**STUDENTER** Joel Ehnroth, Yoonjoo Park
**HANDLEDARE** Pierre Nugues (LTH), Filip Bolling (NordAxon AB)
**EXAMINATOR** Jacek Malec (LTH)

# Korrigering av grammatiska fel med hjälp av maskininlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Joel Ehnroth, Yoonjoo Park**

Verktyg som erbjuder hjälp med stavning och grammatik är viktiga i det digitala samhället. I detta examensarbete presenterar vi maskininlärningsmodeller som presterar bättre än traditionella regelbaserade system.

Detta examensarbete utforskar maskininlärningsmetoder för rättning av grammatiska fel i svensk text, såsom stavfel, böjningsfel, felaktiga ordval, med mera. Resultaten visar att maskininlärningsmetoderna presterar lovvärt och kan korrigera fel till en högre grad än tidigare regelbaserade system.

System för automatisk rättning av olika fel i skriven text har blivit en del av vardagen för många. De återfinns i ordbehandlingsprogram, när vi skriver epost eller SMS, eller när vi gör en sökning via en sökmotor i webbläsaren.

Att kunna formulera sig korrekt och precist är viktigt i dagens samhälle, och att kunna få hjälp av ett verktyg som uppfattar felaktigheter i våra formuleringar är därför av intresse. Särskilt relevant är det möjligen för andraspråkselever som vill underlätta lärandet på egen hand, när en mänsklig pedagog inte finns att tillgå.

Trots att automatiska system för rättning av språkfel har funnits i många år, domineras området av de stora internationella språken, främst engelskan. Detta beror i stor utsträckning på att datatillgång är en central del vid framtagning av maskininlärningsmodeller.

Modellerna som har utvärderats under detta arbete har tränats på stora mängder text från svenska webbsidor. Modellerna kan genom att observera text skapa en intern modell av hur språket fungerar. Denna interna modell kan sedan med fördel användas för en rad olika syften, exempelvis rättning av grammatiska fel.

Gemensamt för maskininlärningsmodeller är dock att de presterar dåligt på uppgifter som de inte specifikt har tränats för. Detta innebär att modellerna ofta måste tränas på ytterligare data som är särskilt avsedd för den specifika uppgiften. I detta fall är uppgiften rättning av grammatiska fel. Att ta fram ett välfungerande system för det svenska språket kräver alltså särskilda resurser för just svensk grammatik, något som i dagsläget är en bristvara. Under arbetets gång utforskas därför möjligheten att använda syntetisk data för att utvidga de tillgängliga resurserna.

Sammanfattningsvis har vi under examensarbetet tagit fram modeller som avgör om en mening är grammatiskt korrekt eller inte, samt modeller som markerar var i meningen fel finns och vilken typ av fel det rör sig om. Till sist utvecklar vi modeller som försöker korrigera felen. Modellerna visar goda resultat och det finns stor potential till vidare utveckling och förbättring.