Master Thesis in Computer Science

Foundational Language Models for Ultra-low Resource Languages
The Case of Faroese

Author

Rói Olsen
rools20@student.sdu.dk

Supervisor:	**Peter Schneider-Kamp** Professor, Dept. of Mathematics and Computer Science University of Southern Denmark
External Supervisor:	**Henrik Hoffmann Nielsen** R&D Responsible, ODIN Ordbogen A/S

June 1, 2025

Department of Mathematics and Computer Science
Faculty of Science, University of Southern Denmark, Campus Odense

© Rói Olsen 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

Abstract

The rapid advancement of large-scale language models has largely benefited high-resource languages, leaving ultra-low resource languages like Faroese underrepresented in natural language processing research. This thesis investigates the feasibility of building foundational language models for Faroese, a North Germanic language spoken by approximately 70,000 people, with limited access to annotated corpora and language technology tools. Focusing on grammatical error correction as a central task, this study explores the development of supporting components including part-of-speech tagging, morphological analysis, lemmatization, and dependency parsing using the spaCy framework and the Faroese Universal Dependencies datasets.
The results show that certain tools, particularly the part-of-speech tagger and morphologizer, achieve promising performance and can serve as a linguistic backbone for downstream tasks. However, other components, such as the lemmatizer and dependency parser, performed poorly due to the limited amount of high-quality annotated data. The grammatical error correction model demonstrated potential but struggled with overcorrection and data quality issues, suggesting that further refinement of training data and corruption methods is needed. A complementary spelling correction model was also explored, showing promise in addressing surface-level orthographic errors.
This work provides practical insights into the challenges and opportunities of building language technologies in data-scarce environments. It highlights the critical role of annotated resources, even in the context of transfer learning, and offers guidance for future natural language processing development in Faroese and other digitally marginalized languages.

Den hurtige udvikling inden for store sprogteknologiske modeller har i høj grad gavnet højressourcesprog, mens ultra-lavressourcesprog som færøsk i vid udstrækning er blevet overset i forskning inden for naturlig sprogbehandling. Denne afhandling undersøger, hvorvidt det er muligt at udvikle grundlæggende sprogteknologiske modeller for færøsk, et nordisk sprog, talt af cirka 70.000 mennesker, som har begrænset adgang til annoterede korpora og sprogteknologiske værktøjer. Med grammatisk fejlkorrigering som det centrale fokusområde udforsker undersøgelse udviklingen af understøttende komponenter såsom ordklassemærkning, morfologisk analyse, lemmatisering og dependensanalyse, baseret på spaCy og de færøske Universal Dependencies-datasæt.
Resultaterne viser, at visse værktøjer, især part-of-speech taggeren og morphologizeren, opnår lovende resultater og kan fungere som et sprogligt fundament for videre analyse. Andre komponenter, såsom lemmatisering og dependency analyse, viste dog ringe præstation på grund af den begrænsede mængde af højtkvalitets annoteret data. Modellen til grammatisk fejlkorrigering viste potentiale, men havde udfordringer med overkorrektion og datakvalitet, hvilket antyder behovet for forbedringer i både træningsdata og fejlindføringsmetoder. En supplerende model til stavekontrol blev også undersøgt og viste lovende resultater i forhold til at rette overfladiske ortografiske fejl.
Dette arbejde giver praktiske indsigter i de udfordringer og muligheder, der følger med

udviklingen af sprogteknologi i miljøer med begrænsede ressourcer. Det fremhæver den afgørende betydning af annoterede datasæt, selv i konteksten af transfer learning, og tilbyder vejledning for fremtidig udvikling af naturlig sprogbehandling for færøsk og andre digitalt marginaliserede sprog.

Table of Contents

List of Figures

List of Tables

List of Graphs

List of Snippets

Glossary

***Bendingargrunnurin* – The Inflection Database** 20

***BERT* – Bidirectional Encoder Representations from Transformers** 2, 7, 13, 29

***bitext* – billingual text** 20

***C4* – Colossal Clean Crawled Corpus** 7

***CNN* – Convolutional Neural Network** 12

***DocBin* – spaCy DocBin** 26

***FFN* – Feed Forward Neural Network** 3

***FN* – False Negative** 8, 31

***FP* – False Positive** 8, 31

***Fróðskaparrit* – Faroese Scientific Journal** 18, 42

***Fróðskapur* – Faroe University Press** 19, 42

***GEC* – Grammatical Error Correction** 1, 9, 10, 11, 12, 13, 21, 28, 47, 49, 50

***GECToR* – Grammatical Error Correction: Tag, Not Rewrite** 10, 11

***GeLU* – Gaussian Error Linear Unit** 7

***GPT* – Generative Pre-trained Transformer** 2, 7, 18

***GT* – Ground Truth** 9

***JSON* – JavaScript Object Notation** 23, 26

***LAS* – Labeled Attachment Score** 34

***LSTM* – Long Short-Term Memory** 2

***Málráðið* – The Faroese Language Council** 20

***mBART* – Multilingual Bidirectional and Auto-Regressive Transformers** 13

***mBERT* – Multilingual Bidirectional Encoder Representations from Transformers** 12, 29

***mC4* – Multilingual Colossal Clean Crawled Corpus** 7

***MORPH* – Morphologizer** viii, 19, 33

***mT5* – Multilingual Text-to-Text Transfer Transformer** ix, 2, 7, 8, 11, 26, 27, 28, 36, 37, 38, 42, 49

***MTD* – The Faroese Centre for Language Technology** 18, 23, 38, 42

***NER* – Named Entity Recognition** 30

***NLLB* – No Language Left Behind** 20, 38

***NLP* – Natural Language Processing** 1, 2, 7, 9, 10, 11, 13, 14, 17, 18, 47, 48, 49, 50, 52

***NUCLE* – Nonnative Corpus of Learner English** 10

***POS* – Part-of-Speech** vi, viii, 1, 9, 11, 12, 17, 19, 25, 26, 29, 30, 31, 32, 33, 41, 47, 51, 52

***ReLU* – Rectified Linear Unit** 7

***REP* – Replacement Pattern** 21

***RNN* – Recurrent Neural Network** 2, 12, 13

***Rættstavarin* – The Spell Checker** 20

***Seq2Seq* – Sequence-to-Sequence** 9, 10, 11, 12, 13

***SNI* – Semantic Textual Similarity** 18

***SOTA* – State-of-the-Art** 2, 13

***T5* – Text-to-Text Transfer Transformer** 6, 7, 13, 17, 18, 26, 27

***TN* – True Negative** 8, 31

***TP* – True Positive** 8, 31

***UAS* – Unlabeled Attachment Score** 34

***UD* – Universal Dependencies** 12, 19, 23, 32

***XLM-R* – Cross-lingual Language Model - RoBERTa** 12

***Eth - byttet ð* – Eth - swapped ð** 40, 43, 46

***Eth - mangler ð* – Eth - missing ð** 40, 43, 46

***Eth - tilføjet ð* – Eth - added ð** 40, 41, 43

Chapter 1

Introduction

The development of large-scale language models has seen remarkable progress in recent years, but this progress has largely been concentrated in high-resource languages. Languages with limited digital presence, often referred to as ultra-low resource languages, remain underrepresented in Natural Language Processing (NLP) research and tools. This thesis investigates the feasibility of developing foundational language models for such languages, with a focus on Faroese. Faroese, spoken by approximately 70,000 people, presents a number of challenges for NLP due to the scarcity of annotated corpora, linguistic tools, and training data. The primary goal of this work is to assess what kinds of language models are realistically achievable for Faroese by surveying the available data and exploring suitable modeling approaches.

Given the constraints of the available resources, this study focuses on developing a Grammatical Error Correction (GEC) model for Faroese, a foundational component for many downstream NLP tasks. To support this goal, a spaCy pipeline will be developed using Faroese Universal Dependencies datasets. The aim is to train a POS tagger, morphologizer, lemmatizer and a dependency parser to support grammatical analysis. This will provide essential linguistic annotations that can enhance the performance of a GEC system. A spelling model will be explored as a complementary resource, aimed at addressing orthographic errors that often co-occur with grammatical mistakes.

This thesis aims to contribute a practical perspective on the development of core language models in settings where data is scarce, providing insights into model performance, data requirements, and potential paths forward for NLP in ultra-low resource language contexts.

Chapter 2

Scientific Background

This chapter provides an overview of the background information necessary to understand the context of the study. It covers the key concepts and relevant research that have shaped the current state of the field.

2.1 Transformers

Transformers are a class of deep learning models introduced by (Vaswani et al., 2017) that revolutionized NLP and other sequential data tasks. Unlike Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM), which process input sequentially, transformers leverage a self-attention mechanism that allows for parallelization and long-range dependency modeling. At the core of the transformer architecture is the self-attention mechanism, which enables the model to weigh the importance of different words in a sequence, regardless of their position. This is complemented by positional encodings, which provide information about word order, addressing the lack of inherent sequentiality in self-attention. The transformer is composed of stacked encoder and decoder layers, each containing multi-head self-attention and feedforward layers with residual connections and layer normalization. Self-attention will be covered in more detail in Section 2.1.1.
Transformers have been the foundation for State-of-the-Art (SOTA) NLP models, including Bidirectional Encoder Representations from Transformers (BERT), mT5 and perhaps the best known transformer to the general public, Generative Pre-trained Transformer (GPT). These architectures have been widely applied in text generation, translation, and classification tasks, as well as in domains such as computer vision, protein structure prediction, and reinforcement learning. The scalability and effectiveness of transformers have driven their adoption across various fields, making them a cornerstone of modern deep learning research.

The original Transformer architecture proposed by (Vaswani et al., 2017) is built around an encoder-decoder structure: the encoder processes the input sequence, while the

decoder generates the output. Both the encoder and decoder are composed of stacks of identical layers, six in the original model. Each encoder layer consists of two main components: a multi-head self-attention mechanism and a Feed Forward Neural Network (FFN), each followed by a residual connection and layer normalization. Before entering the encoder, input tokens are passed through an embedding layer and enriched with positional encodings to inject information about the sequence order. Each decoder layer includes three subcomponents: a masked multi-head self-attention layer, to prevent attending to future tokens, a multi-head attention mechanism over the encoder outputs, and a FFN. Like the encoder, each of these sublayers is followed by a residual connection and layer normalization. A visual overview of this architecture is provided in the diagram at Figure 1.

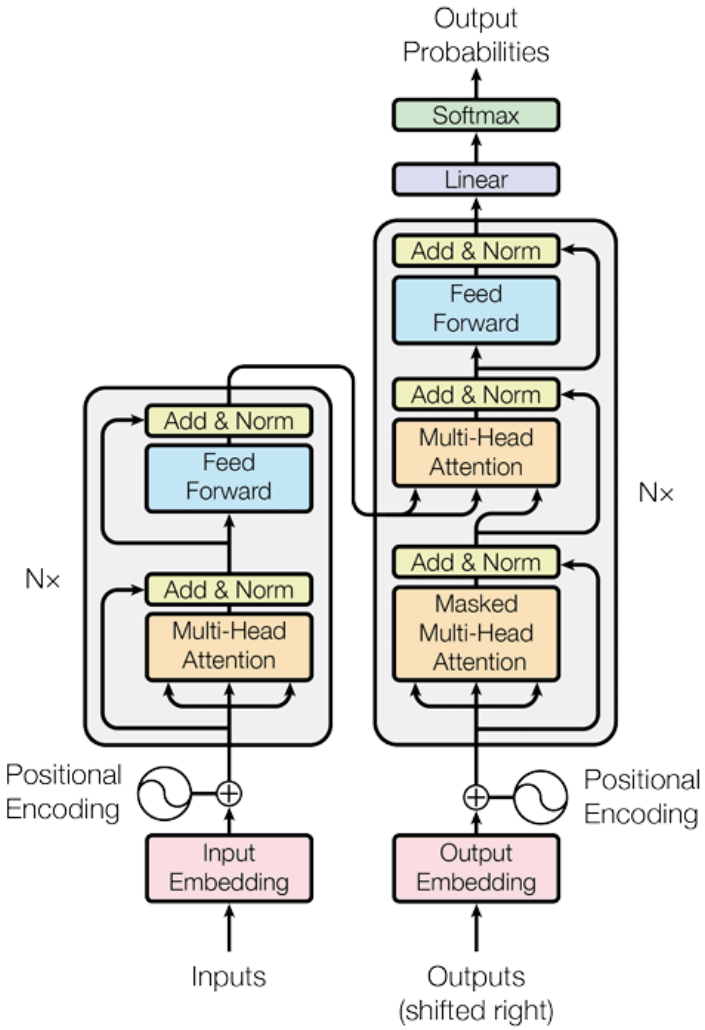


Figure 1: The original transformer from "Attention Is All You Need". (Vaswani et al., 2017)

2.1.1 Self-attention

Self-attention is a mechanism that allows a model to weigh the importance of different words in a sequence when making predictions. It computes a weighted sum of the input representations, where the weights are determined by the similarity between the words. This enables the model to capture long-range dependencies and relationships between

words, regardless of their position in the sequence. Self-attention is a key component of transformer architectures, enabling them to process sequences in parallel and learn contextual representations effectively. Self-attention is a sequence to sequence process that takes sequence of vectors as input, these can be called $x_1, x_2..., x_t$, and outputs a sequence of vectors $y_1, y_2..., y_t$. The dimentionality of the input and output vectors is the same. To produce the output vector y_i, the self-attention operation computes a weighted sum of the input vectors, where the weights are determined by the similarity between the input vectors. The self-attention operation can be expressed mathematically as follows:

$$y_i = \sum_j w_{ij} x_j \tag{1}$$

Where j indexes over the input vectors, and the weights sum up to 1. w_{ij} is a computed value, and not a parameter. The weights are computed using a similarity function, which measures the similarity between the input vectors. The most common similarity function used in self-attention is the dot product. which is computed as follows:

$$w'_{ij} = x_i^\top x_j \tag{2}$$

The dot product gives a result that is in the range $[-\infty, \infty]$. So to normalize the weight, a softmax functioon is applied to the weights, which transforms the weights into a probability distribution. The softmax function is defined as follows:

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}} \tag{3}$$

That is the basic self-attention operation, but it is not the only one. On Figure 2 is an illustration of the self-attention mechanism.

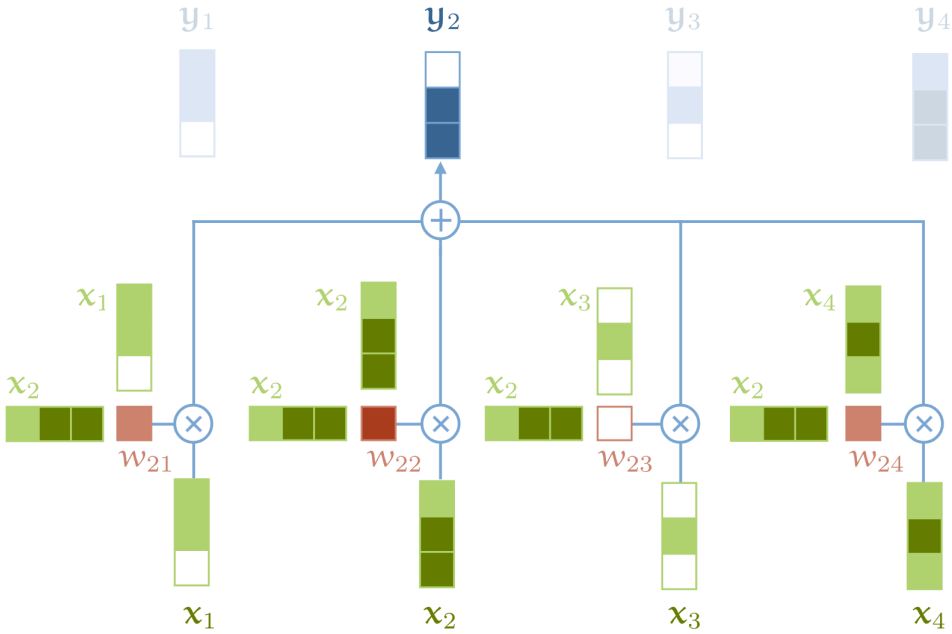


Figure 2: Self-attention mechanism illustrated. Not including the softmax operation. Credit to Peter Bloem

Transformers use a more complex version of self-attention. The input vector x_i is used in 3 different ways. It's compared to every other input vector to compute its own output y_i. It's also compared to every other input vector to compute the output of every other input vector y_j. And lastly it is also used as a part of the weighted sum to compute the output of each output vector, once the weights have been computed. These roles are called **query**, **key**, and **value**. To compute the query, key, and value vectors for a given input vector x_i, three separate linear transformations are applied using learned weight matrices W_q, W_k, W_v, each of size $k \times k$. These matrices project the input into three different representations:

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i \tag{4}$$

$$w'_{ij} = q_i^\top k_j \tag{5}$$

$$w_{ij} = \text{softmax}(w'_{ij}) \tag{6}$$

$$y_i = \sum_j w_{ij} v_j \tag{7}$$

The softmax function is sensitive to the scale of its input values. To address this, a scaling factor of $\sqrt{k}$, where k is the dimensionality of the key vectors, is introduced. This scaling helps stabilize gradients and improves convergence during training. The scaled dot-product attention is defined as:

$$w'_{ij} = \frac{q_i^\top k_j}{\sqrt{k}} \tag{8}$$

The rationale behind this scaling is rooted in how vector magnitudes behave in high-dimensional spaces. For example, a vector in $\mathbb{R}^k$ where every element is c has a Euclidean length of $c\sqrt{k}$ As a result, dot products tend to grow with k, potentially producing large values that cause the softmax to become sharply peaked, making it overly confident in just a few positions. Dividing by $\sqrt{k}$ normalizes this effect, keeping the scale of the dot products consistent across different dimensions.

Another important aspect of self-attention is the use of multiple attention heads. Instead of computing a single set of attention weights, multiple sets are computed in parallel, each with its own learned weight matrices. This allows the model to capture different types of relationships and dependencies in the data. The way this is done is by splitting the input vector into h different parts, where h is the number of attention heads. Each part is then used to compute a separate set of query, key, and value vectors. The outputs from all attention heads are concatenated and passed through a final linear transformation to produce the final output. This multi-head attention mechanism allows the model to learn richer representations by attending to different parts of the input sequence simultaneously. The multi-head attention mechanism can be expressed mathematically as follows:
If you have:

- An input matrix $X \in \mathbb{R}^{n \times d_{\text{model}}}$, where n is the number of tokens and d_{model} is the dimension of each token vector.
- h attention heads.

For each attention head r:

1. Project the input into the query, key, and value vectors using separate learned weight matrices:

$$Q = XW_q^r \quad K = XW_k^r \quad V = XW_v^r \tag{9}$$

where $W_q^r, W_k^r, W_v^r \in \mathbb{R}^{d_k \times d_{\text{model}}}$ (often $d_k = \frac{d_{\text{model}}}{h}$).

2. compute the scaled dot-product attention for each head:

$$\text{Attention}^r = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{10}$$

3. Concatinate the outputs of all heads:

$$\text{Concat}(\text{head}_1, ..., \text{head}_h) \tag{11}$$

4. Pass the result through a final linear projection:

$$\text{MultiHead}(Q, K, V) = \text{Concatinate}(\text{head}_1, ..., \text{head}_h)W_o \tag{12}$$

where W_o is the Final projection matrix that maps concatenated head outputs back to d_{model}

2.1.2 T5

Building on the foundation laid by the original Transformer architecture introduced in "Attention Is All You Need", the Text-to-Text Transfer Transformer (T5) model proposed by (Colin Raffel et al., 2019) marks a major advancement in the unification and simplification of natural language processing tasks. It introduces a flexible and scalable framework that reframes all NLP problems within a single, consistent text-to-text format. In this setup, both inputs and outputs are treated as strings, regardless of the task. For instance, translation is approached as:
"translate English to German: That is good → Das ist gut"
This unified formulation eliminates the need for task-specific heads or output formats, streamlining architecture design and enabling more effective multitask and transfer learning.
Beyond its task framing, T5 incorporates several architectural innovations. To improve generalization on sequences of varying length, it replaces absolute positional embeddings with relative positional bias, allowing the model to focus on the relative distances between tokens rather than fixed positions. In terms of normalization, it departs from the post-layer norm approach used in the original Transformer, opting instead for pre-layer normalization, which applies layer norm before the attention and feed-forward

sub-layers, an adjustment that enhances training stability, especially in deeper variants. Dropout on attention weights, commonly used in earlier models, is also disabled here, as it was shown to negatively impact performance during large-scale pretraining. In the feed-forward layers, the standard Rectified Linear Unit (ReLU) activation is replaced with Gaussian Error Linear Unit (GeLU), offering smoother gradients and better performance in deeper networks.
For tokenization, the model leverages a byte-level SentencePiece tokenizer, which avoids the issue of unknown tokens and supports improved handling of multilingual and non-standard text by operating directly on raw byte sequences.
When it comes to pretraining, T5 introduces a distinct span-corruption objective. Unlike BERT's masked token prediction or GPT's autoregressive formulation, this approach involves masking out random spans of text and replacing them with sentinel tokens (e.g., <extra_id_0>), tasking the model with reconstructing the missing spans. This span-level corruption helps the model develop stronger contextual understanding and more flexible generative capabilities.
Finally, the model family was released in multiple scales from T5-Small to T5-XXL and trained on the Colossal Clean Crawled Corpus (C4) dataset. Its design emphasizes scalability, demonstrating that performance improves predictably with increased model and data size, making it a cornerstone example of scaling laws in NLP.

2.1.3 mT5

Building on the architecture and pretraining paradigm established by T5, the mT5 model by (Linting Xue et al., 2021) extends these principles to the multilingual domain. It retains the same encoder-decoder structure and self-attention mechanisms but is redesigned to perform well across a wide range of languages. This shift introduces several key adaptations to enhance cross-lingual generalization and mitigate the limitations of English-focused training.
mT5 is trained from scratch on a large, diverse multilingual corpus, incorporating improvements to tokenization and training strategy. Instead of T5′s subword tokenizer, it uses a SentencePiece unigram model trained on data from 101 languages, boosting its handling of non-English scripts. It also departs from T5′s loss computation method by including padding tokens in gradient calculations, promoting greater robustness in multilingual settings.
Unlike English-only pretrained models, it leverages the multilingual Multilingual Colossal Clean Crawled Corpus (mC4) corpus from the outset, enabling broader language coverage. Additionally, it avoids language tags or specialized embeddings, learning to interpret inputs based purely on content. While the core architecture mirrors that of T5, these targeted modifications make mT5 substantially more effective for global NLP.

2.2 Evaluation Metrics

This study uses the evaluation metrics **precision**, **recall**, **accuracy**, and the $\mathbf{F}_\beta$ score. These are commonly applied in NLP tasks to assess a model's ability to correctly identify

relevant instances. Each prediction falls into one of four categories: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). These values are used to compute the evaluation metrics.

The intuition behind each metric is as follows:
Precision When the model predicts a positive, how often is it correct?
Recall When a positive instance exists, how often does the model correctly identify it?
Accuracy The overall rate of correct predictions, considering both positive and negative classes.
$\mathbf{F}_\beta$ A single score that balances precision and recall. The weighting depends on the value of β:
$\beta = 1$ precision and recall are weighted equally ($\mathbf{F}_1$ score).
$\beta < 1$ precision is weighted more heavily.
$\beta > 1$ recall is weighted more heavily.

The formulae for the metrics are:

$$\textbf{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

$$\textbf{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

$$\textbf{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{15}$$

$$\mathbf{F}_\beta = (1 + \beta^2) * \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}} \tag{16}$$

The way TN is counted is non-standard for the mT5 models, as it it only counted, when a whole sentence is correctly predicted to not contain any errors.

In addition to these metrics, hit, correct, and incorrect rates are also used. The names come from internal tools at Ordbogen. They are defined as follows:
Hit rate The percentage of instances where the model correctly identifies the correct error type.
Correct rate The percentage of instances where model correctly identifies the error and corrected it. (TP + TN)
incorrect rate The percentage of instances where the model made a prediction but it was not correct. (FP)
These rates provide additional insights into the model's performance, particularly in terms of its ability to correctly identify and correct errors. The correct rate is a percentage of how many of the total errors were corrected, while the incorrect rate is how many were incorrectly predicted. This means if an error type has a correct of 20% and an incorrect of 10%, 20% of the errors in the testset were corrected and 10% of the errors were incorrectly predicted and 70% were not detected and therefore unchanged. While these metrics may be non-standard, it helps in giving insight into how good the

model is at predicting a specific error, when a test contains only one error, this can then be suplemented the other metrics to get an overall picture.

2.3 Cross-entropy Loss

Cross-entropy loss is one of the most used loss functions in NLP tasks, especially in tasks such as POS tagging, dependency parsing and GEC. It measures the difference in two probability distributions: the Ground Truth (GT) and the distribution predicted by the model. Formally, cross-entropy loss is defined as:

$$L = -\sum y_i \log(\hat{y}_i) \tag{17}$$

The GT is often represented as a one-hot encoded vector, this means that a vector of size n is created, where n is the number of classes or categories, and one element in the vector is set to be "hot", i.e., it is set to 1, while all the other elements are "cold", i.e., set to 0. So the cross-entropy loss can be simplified to:

$$L = -\log(\hat{y}_k) \tag{18}$$

Where k is the index of the "hot" or correct element in the GT vector. This formula penalizes the model more heavily for being confident in an incorrect prediction, as it will result in a lower probability for the correct class, leading to a higher loss, and it also encourages the model to be more confident in its correct predictions.
In the context of sequence tagging and Sequence-to-Sequence (Seq2Seq) tasks, such as GEC, cross-entropy loss is typically computed at each time step of the output sequence. The total loss is then the sum or average over all positions in the sequence. This makes cross-entropy especially suitable for token-level prediction tasks, where each token is assigned a categorical label, such as, a corrected word, a POS tag, or an edit operation such as KEEP or REPLACE. One of the key strengths of cross-entropy loss is its strong theoretical grounding in information theory. It measures the average number of bits needed to identify an event from a set of possibilities, given a predicted distribution. As such, minimizing cross-entropy is equivalent to maximizing the likelihood of the correct labels, which aligns naturally with the goal of most probabilistic classifiers and generative models. Cross-entropy loss is a standard objective for classification tasks, but it can struggle with class imbalance, a common issue in GEC datasets where the majority of tokens are labeled as KEEP. This skewed distribution can cause the model to favor the majority class, reducing its ability to detect actual errors. To mitigate this issue, researchers have proposed modifying the loss function to account for the unequal distribution of labels. For instance, (Trong Huy Phan & Kazuma Yamamoto, n.d.) show that cross-entropy loss, when applied in the context of object detection, underperforms for underrepresented classes due to its uniform treatment of class frequencies. They explore weighted variants such as Balanced Cross-Entropy and Focal Loss, which can be adapted to sequence tagging tasks like GEC to emphasize the minority classes and enhance model robustness under imbalance. Another challenge associated with cross-entropy loss is its tendency to make models overconfident in their predictions. This is

particularly problematic in low-resource settings, where training data is limited and the model may overfit. A widely adopted solution is label smoothing, a regularization technique that softens the target probability distribution. Instead of assigning a probability of 1.0 to the correct label and 0.0 to all others, label smoothing assigns a slightly lower probability, such as 0.9, to the correct label and distributes the remainder uniformly across the others. This prevents the model from becoming overly confident and helps improve generalization. (Li Guo et al., n.d.) provide both empirical and theoretical evidence for the effectiveness of label smoothing as a regularization method. Their work connects label smoothing with the Neural Collapse phenomenon and demonstrates that it accelerates convergence and improves model calibration in deep learning settings, supporting its relevance even in tasks beyond standard classification.

2.4 Grammar Error Correction

GEC is the task of automatically detecting and correcting grammatical errors in written text. It is typically treated as a form of text-to-text transformation, where the goal is to convert grammatically incorrect input into a corrected version that adheres to the norms of the target language (Christopher Bryant et al., n.d.; Siew Mei Ng & Hwee Tou Ng, n.d.). GEC systems are designed to address a wide range of error types, including but not limited to:

- Morphological errors (e.g., incorrect verb conjugation or noun inflection)
- Syntactic errors (e.g., subject-verb disagreement, word order issues)
- Orthographic and spelling errors
- Punctuation errors
- Lexical choice errors

The task was initially motivated by applications in second language learning and writing assistance (Claudia Leacock et al., n.d.), but has since become relevant for broader NLP tasks, such as preprocessing noisy text in user-generated content or learner corpora for use in machine translation, summarization, and other downstream models. Approaches to GEC have evolved from rule-based (Alla Rozovskaya & Dan Roth, n.d.; Martin Chodorow et al., n.d.) and statistical methods (Chris Brockett & William B. Dolan, n.d.) to neural network-based models, especially those based on the Transformer architecture (Vaswani et al., 2017). In many modern systems, GEC is framed as either a Seq2Seq problem, similar to machine translation (Jeffery Lichtarge et al., n.d.; Roman Grundkiewicz & Marcin Junczys-Dowmunt, n.d.), or as an edit-based classification task, where the model predicts a sequence of edits to transform the input into a correct output. The latter approach, exemplified by Grammatical Error Correction: Tag, Not Rewrite (GECToR) (Kostiantyn Omelianchuk et al., n.d.), has gained popularity due to its efficiency and strong performance on limited data. The performance of GEC systems depends heavily on the availability of annotated corpora such as Nonnative Corpus of Learner English (NUCLE) (Daniel Dahlmeier et al., n.d.) or Lang-8 (Tomoya Mizumoto et al., n.d.) , which are largely limited to high-resource languages like English. This poses a significant barrier for low- and ultra-low-resource languages, where high-

quality error-annotated corpora are rare or nonexistent. To address this, researchers have explored cross-lingual transfer (Ikumi Yamashita et al., n.d.), data augmentation using synthetic errors (Ziang Xie et al., n.d.), and multilingual pre-trained models like mBART (Yinhan Liu et al., n.d.) and mT5 (Linting Xue et al., 2021). For this study , the focus is on a Seq2Seq approach to GEC.

2.4.1 Sequence Tagging

Sequence tagging is a common approach in NLP tasks, where the goal is to assign labels to each element in a sequence. This can be applied to various tasks, such as POS, named entity recognition, and GEC. In the context of GEC, sequence tagging involves predicting a sequence of edits that transform the input into a grammatically correct output. This approach allows for efficient processing and can leverage existing pre-trained models to improve performance on low-resource languages. Instead of rewriting the entire sentence, the model tags each token or span of tokens with a label. The labels will often fall into one of the following categories:

- **KEEP**: The token is correct and should be kept as is.
- **DELETE**: The token is incorrect and should be removed.
- **REPLACE**$_{\textbf{token}}$: A token should be replaced with a the specified token.
- **APPEND**$_{\textbf{token}}$: The specified token should be appended after this position.

These edit tags are typically learned through supervised training on annotated datasets, where each token in an erroneous sentence is aligned with its corresponding edit in the corrected sentence. Models like GECToR take this approach, using a transformer-based architecture fine-tuned on GEC tasks. By treating GEC as a sequence tagging problem rather than a generation problem, these models can be both more efficient and more interpretable. One major advantage of this approach is its ability to produce high-quality corrections with relatively small amounts of data, making it particularly well-suited for ultra-low-resource languages like Faroese. Additionally, the tagging process ensures a strong alignment with the original sentence, which helps preserve the writer's intent and style. This is especially important in applications like writing assistance, where maintaining the original meaning while correcting errors is crucial. The downside of this approach is that more complex errors, that require multiple edits or reordering of tokens, may be harder to capture. This means that problems like rephrasing or restructuring sentences may not be as effectively handled by a sequence tagging model.
While this study adopts a Seq2Seq approach, sequence tagging remains relevant as a framework for structuring training data. Inspired by tagging-based annotations, the data includes aligned edits with associated error types, which not only guide the Seq2Seq model during training but also enable user-facing feedback by identifying the nature of each correction.

Part of Speech Tagging & Morphological Analysis

POS tagging and morphological tagging are foundational tasks in NLP, providing structured linguistic annotations that support downstream applications such as syntactic parsing, named entity recognition, and GEC. For morphologically rich and ultra-low-resource languages like Faroese, accurate tagging is especially important, as

key grammatical information is often expressed morphologically rather than through fixed word order (*Universal Dependencies V2: An Evergrowing Multilingual Treebank Collection*, 2020). POS tagging assigns each token in a sentence a label that indicates its syntactic category, such as NOUN, VERB, or ADJ. These categories define the functional role of the word in a sentence and are typically drawn from a tagset such as the Universal POS tagset used in the Universal Dependencies (UD) project (Slav Petrov et al., 2012). These standardized tags help ensure consistency across languages, enabling cross-linguistic comparison and model transfer. Morphological tagging provides a more fine-grained analysis by assigning grammatical features to tokens, including number, gender, case, tense, mood, definiteness, and others. Each token may receive multiple morphological labels, forming a detailed grammatical profile. Unlike POS tagging, morphological tagging is thus a multi-label classification task, where each token may carry a set of attribute-value pairs, such as Tense=Past, Number=Plur, or Case=Acc (*A Gold Standard Dependency Corpus for English*, 2014). Both POS and morphological tagging are commonly treated as sequence labeling problems. Given a sequence of words, the model must predict a sequence of corresponding labels. Neural approaches dominate current systems, using architectures such as Convolutional Neural Network (CNN)s, RNNs, or transformers to build contextualized token representations. These representations are then passed to task-specific classifiers. In spaCy, the tagger component predicts POS tags, while the morphologizer predicts morphological features using multi-label classification. The morphologizer component in spaCy learns from structured annotations and attempts to predict multiple features independently, although joint feature modeling is also possible. Morphological predictions can be enhanced by dictionaries, rule-based systems, or constraints derived from morphological paradigms (Reut Tsarfaty et al., 2010). In low-resource settings, where annotated corpora are scarce, transfer learning, cross-lingual projection, and data augmentation techniques have proven useful. Pretrained multilingual models, such as Multilingual Bidirectional Encoder Representations from Transformers (mBERT) or Cross-lingual Language Model - RoBERTa (XLM-R), can transfer knowledge across languages with shared syntactic or morphological properties (*Unsupervised Cross-Lingual Representation Learning at Scale*, 2020). Additionally, rule-based systems or lookup tables, such as those derived from Wiktionary or finite-state transducers, can be used to bootstrap annotations or support training in the absence of large datasets.

2.4.2 Sequence-To-Sequence Grammatical Error Correction

Seq2Seq models are a widely adopted approach in GEC, offering an end-to-end architecture that transforms gramatically incorrect input text into grammatically correct output. These models, inspired by machine translation, use an encoder-decoder framework to learn how to map error-prone text to its corrected form, often incorporating attention mechanisms or transformer architectures to enhance performance. The encoder processes the input sentence and encodes it into a latent representation, which the decoder then uses to generate a corrected sentence token by token. This design enables Seq2Seq models to capture a broad range of grammatical corrections, including insertions, deletions, substitutions, and reordering. Early work in this area,

such as by (Zheng Yuan & Ted Briscoe, n.d.), showed that neural machine translation methods could be effectively adapted for grammatical error correction. The transition to attention-based and transformer-based models led to substantial improvements. For example, (Marcin Junczys-Dowmunt et al., n.d.) demonstrated that grammatical error correction could be framed as a low-resource translation task and achieved competitive results using the MarianNMT framework. Pre-trained transformer models such as BERT, T5, and Multilingual Bidirectional and Auto-Regressive Transformers (mBART) have further pushed the field forward. (Masahiro Kaneko & Mamoru Komachi, n.d.) demonstrated that encoder-decoder models can learn syntactic structures relevant to GEC, while (Jeffery Lichtarge et al., n.d.) proposed synthetic data generation techniques that significantly reduced the reliance on annotated corpora. One of the strengths of Seq2Seq models is their flexibility. They are not restricted to a fixed set of edit operations, enabling them to produce fluent and coherent corrections. However, this comes with potential drawbacks: such models may overcorrect, changing tokens unnecessarily, or introduce semantic divergence by prioritizing fluency over fidelity to the original meaning (Roman Grundkiewicz & Marcin Junczys-Dowmunt, n.d.). Despite these challenges, Seq2Seq models continue to be central in SOTA GEC systems, especially when combined with multilingual pre-training and synthetic error generation methods such as those proposed by (Ziang Xie et al., n.d.) and (Yinhan Liu et al., n.d.).

2.4.3 Dependency Parsing

Dependency parsing is a fundamental task in NLP that involves analyzing the grammatical structure of a sentence by establishing relationships between "head" words and their dependents. This syntactic representation results in a dependency tree, where nodes correspond to individual words and directed edges define grammatical relationships. Dependency parsing is particularly well-suited for morphologically rich and free word-order languages, as it emphasizes relations between words rather than phrase boundaries(Joakim Nivre, 2015). Dependency parsing differs from constituency parsing in that it models binary relationships directly between words, offering a more compact and often more linguistically intuitive structure. Each word in the sentence (except the root) is linked to another word, its syntactic head, forming a tree rooted at the main verb or clause head. This structure aligns closely with how language is processed in many downstream applications (Daniel Jurafsky & James H. Martin, 2021). Various algorithms have been developed for dependency parsing:

- **Transition-Based Parsers** incrementally build dependency trees using a series of actions (such as shift, left-arc, and right-arc). These parsers are fast and suitable for real-time applications, but may be limited in handling non-projective dependencies (Matthew Honnibal et al., n.d.).
- **Graph-Based Parsers** consider all possible trees and select the one with the highest score. These parsers are generally more accurate and capable of handling non-projective dependencies but are computationally more expensive (Ryan McDonald et al., 2005).
- **Neural Network-Based Parsers** represent the current SOTA, employing RNNs, attention mechanisms, or transformer architectures. For instance, Dozat and

Manning's biaffine parser significantly improved performance across multiple treebanks by incorporating attention-based scoring of arcs and labels (Timothy Dozat, 2017).

Dependency parsing has wide-reaching applications in modern NLP systems:

- **Machine Translation** Parsing improves the alignment and generation quality by providing syntactic structure information for reordering and disambiguation (Daniel Jurafsky & James H. Martin, 2021).
- **Information Extraction** Relationships between entities such as subjects and objects can be directly identified using dependency links.
- **Question Answering** Parsing enhances understanding of question structure and improves answer extraction.
- **Sentiment Analysis** Dependency paths help determine which adjectives or modifiers apply to which entities.

The use case for a dependency parcer in this study is **Information Extraction**. The dependency parser will help to identify certain grammatical structures and relationships, so that that can be used to make sure that the required context for an error is present. For example, verb tense disagreement errors can be introcuded by detecting the dependencies in the sentence. A sentence like "She walked into the room and saw the mess." can be used to introduce a verb tense disagreement error by changing the verb "walked" to "walks", resulting in "She walks into the room and saw the mess." and to make this error, it is important to be sure that such a dependency structure is present, if a the tense in the sentence "She said that she was tired." is changed, it just changes the meaning slightly, but the sentence is still grammatically correct, so without a dependency parser, it is difficult to identify the correct cases to corrupt.

2.5 Spelling

There are 2 categories of spelling errors, the first is a typographical error, the second is a cognitive error. Typographical errors are made my mistyping a word, while cognitive errors are made by a lack of understanding. A spelling error in the context of this study is an error that results in a word that is not in the faroese dictionary.

2.5.1 Typographical Errors

Typographical errors can be split into 4 subcategories:
Transposition errors can be made by transposing two adjacent characters in a word.
Substitution errors can be made by substituting a character in a word.
Insertion errors can be made by inserting a character into a word.
Omission errors can be made by omitting a character in a word.

Omission errors are the simplest to make, since they only require removing a character from the original word. Transposition errors are also very simple to make, since they only require permutation of the original word, by swapping a pair of adjacent characters.

Substitution and Insertion errors require a bit more thought. If characters are picked completely at random, most errors would not be realistic typos and it would result in most generated spelling errors being redundant. For a letter like **G** only 6/29 typos would be useful and a letter like **Q** would only have 3/29 useful errors. To make the errors more realistic, each character that is substituted or inserted is picked from adjacent characters on the keyboard. For example replacing a **G** with an **H** is a probable typo to make when writing on a keyboard, since the h key is right next to the **G** key. The same goes for inserting a character, if a character is inserted, it is picked from the adjacent characters on the keyboard. The keyboard layout used is the faroese keyboard layout, which is a modified version of the danish keyboard layout. The only relevant difference between the two layouts is that the faroese keyboard layout has the **Ð** character to the right of **Å**. The other difference is in punctuations and modifiers.

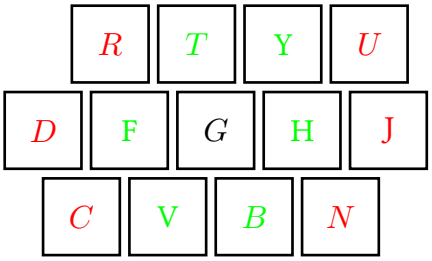


Figure 3: Possible typographical errors that can be made for G are marked with green and the red are too far away from the correct key to be considered a probable typo.

2.5.2 Cognitive Errors

Cognitive spelling errors are generally more difficult to categorize than typographical ones, as their causes are often linked to underlying linguistic knowledge, perception, or variation. Many of these errors are phonetic in nature, arising when words are spelled based on how they sound to the writer, but other cognitive patterns also play a role.
A frequent source of phonetic errors in Faroese involves the letter **ð**. Since **ð** is typically silent, writers often omit it entirely. In cases where it is pronounced, it may sound like a **v**, which leads to substitution errors where **v** is used in place of **ð**. Another relevant pattern is the use of **g** in contexts like hagan, where **g** is pronounced more like a **j**. In these cases, **g** plays a similar phonological role to **ð**, serving to break up vowel clusters, somewhat like an apostrophe in English. This can cause confusion in spelling, where writers may replace it with **ð**, reflecting its pronunciation rather than its correct form.
Dialectal variation further contributes to phonetic spelling errors. Faroese dialects differ significantly, and these differences often influence spelling. For instance, speakers from northern dialect regions such as Klaksvík pronounce **á** as **a**, leading to spelling errors where **á** is replaced with **a**. In contrast, speakers from southern areas may articulate **p** more softly, which can result in it being written as **b**.
In addition to phonetic influences, many cognitive errors stem from the misapplication of spelling rules. A common rule in Faroese states that short vowels are typically followed by double consonants, while long vowels precede single consonants. However, there are exceptions, such as **skula** and **fram**, where the vowels are short but followed by only

one consonant. These exceptions can cause learners to overapply the general rule and write **skulla** or **framm**, which are phonologically logical but orthographically incorrect. These cognitive spelling errors, whether rooted in phonetics, dialect, or rule-based overgeneralization, highlight the linguistic complexity faced by Faroese writers. Understanding these patterns is crucial for developing effective spelling correction models that can distinguish between common cognitive slips and truly unexpected forms.

Chapter 3

Materials

This chapter describes the materials used in the study, including the datasets, tools, and resources that were essential for conducting the experiments and analyses.

3.1 spaCy

Spacy is an open-source software library for advanced NLP in Python. It is designed specifically for production use and is widely used in industry. Spacy provides a wide range of features for processing text data, including tokenization, POS tagging, named entity recognition, and dependency parsing. It also includes pre-trained models for a variety of languages and domains, making it easy to get started with NLP tasks. The relevant features of spaCy for this thesis are the POS tagger and morphologizer, lemmatizer and dependency parser.

3.2 NanoT5

The NanoT5 framework, introduced by (Piotr Nawrot, 2023), addresses the computational challenges associated with training large-scale Transformer models like T5. Recognizing that the substantial resource requirements of such models can hinder research accessibility, NanoT5 offers a streamlined, PyTorch-based solution for efficient pre-training and fine-tuning of T5-style models, particularly under constrained computational budgets. The primary motivation behind NanoT5 is to democratize access to advanced language modeling by reducing the computational barriers to training T5 models. While existing implementations of T5 often rely on resource-intensive frameworks like JAX/Flax, NanoT5 provides a PyTorch-based alternative that emphasizes simplicity and efficiency. By optimizing various components of the training pipeline, NanoT5 enables researchers to pre-train a T5-Base model on a single GPU within approximately 16 hours, without compromising performance. The key features of NanoT5 include:

- **On the fly data processing**
- **Flexible optimizer support** The original Adafactor optimizer is supported, but the framework also allows for the use of other optimizers like AdamW.
- **Fine-tuning** NanoT5 supports fine-tuning on downstream tasks, enabling users to adapt pre-trained models to specific applications. The framework has been applied to Semantic Textual Similarity (SNI) benchmark, demonstrating its effectiveness in fine-tuning tasks.
- **Accessibility and reproducibility** NanoT5 is open-source and available on Github, promoting transparency and reproducibility in NLP research. The framework is designed to be user-friendly, with clear documentation and examples to facilitate adoption by the research community.

The training objective of the framework is the same as the original T5 model, which is span masking. This means that during training, the model learns to predict masked spans of text within a sequence, allowing it to capture contextual information and relationships between words effectively. The version of NanoT5 used in this study is a fork of the original repository by (*Schneiderkamp Lab Github Repository*, n.d.).

3.3 Language Resources From MTD

The Faroese Centre for Language Technology (MTD) Has released a number of resources for the Faroese language. A Github repository (*Faroese Language Resources by Fonlp*, n.d.) contains a collection of datasets and models (*Faroese Language Resources by the Centre for Language Technology*, n.d.).

A private corpus developed by Uni Johannesen was used as a testset to evaluate the performance of the grammar and spelling models. The corpus contains ~2700 sentences and consists of manually corrected and annotated essays from students. The essays were written in Faroese and contain a variety of errors, including spelling, grammar, and punctuation errors. The testset was originally developed for a thesis to evaluate a GPT−4o model on how well it understands faroese grammar and correct errors. Additionally the data used in the study by (Katrin Næs, 2005) on spelling errors in faroese students' essays, was also used in this study. In addition, a private high quality corpus of faroese sentences was provided by MTD for training.

3.4 Faroe University Press Papers And Books

Faroese Scientific Journal (Fróðskaparrit) has published a number of papers and books in the various fields. The file format of the papers and books is pdf, so some preprocessing is needed to get the text out of them. All publications can be found for free on their website. The papers are from Fróðskaparrit. Fróðskaparrit is an annual journal with scientific articles from and about the Faroe Islands and Faroese issues. The journal spans all scientific fields with articles in Faroese (mostly Humanities and Social Sciences)

or English (Natural and Life Sciences). The books are from Faroe University Press (Fróðskapur). Fróðskapur was established in 2005. It publishes works from all scientific fields in the Faroe Islands. Publications are in Faroese, English and Danish. For this study only the faroese publications have been used.

3.5 Universal Dependencies

The two Faroese UD datasets used in this study are the (*Faroese-OFT Treebank*, n.d.) and (*Faroese-Farpahc Treebank*, n.d.) treebanks. Both datasets consist of manually annotated Faroese sentences.

(*Faroese-OFT Treebank*, n.d.) is based on sentences from the Faroese Wikipedia. The entire Wikipedia was processed using (*Trond Trosterud's Tools for Faroese*, n.d.), and sentences containing unknown words were removed. The remaining sentences were manually annotated for Universal Dependencies. Morphological and POS tags were converted deterministically using a lookup table, and errors in the original morphology and disambiguation were corrected where found. As is typical for Wikipedia-based corpora, the dataset contains a high proportion of copular constructions and relatively few first- and second-person forms. Additionally, the quality of the original Wikipedia articles may vary, which can impact the consistency and reliability of the syntactic structures found in the data.

(*Faroese-Farpahc Treebank*, n.d.) is a conversion of the (*Faroese Parsed Historical Corpus (Farpahc)*, n.d.) corpus to the Universal Dependencies scheme using (*Udconverter*, n.d.). (*Faroese Parsed Historical Corpus (Farpahc)*, n.d.) is a 53,000-word corpus consisting of three texts from the 19th and 20th centuries, originally manually parsed according to the (*Penn Parsed Corpora of Historical English*, n.d.) annotation scheme. Two of these parsed texts were automatically converted to the UD format, resulting in the 40,000-word (*Faroese-Farpahc Treebank*, n.d.) treebank. Due to the historical nature of the source material, the grammar and vocabulary may diverge significantly from contemporary Faroese, which could limit the dataset's applicability to modern language use.

The files with POS and MORPH labels contain 6,652 Faroese sentences, totaling 9.3 MiB of data. The lemma-labeled files include 1,428 sentences (756 KiB), while the files annotated with dependency parser labels comprise 3,049 sentences, amounting to 2.8 MiB.

3.6 Bendingar.fo

Bendingar.fo is a website, that hosts The Inflection Database (Bendingargrunnurin) (*Bendingarunnurin*, n.d.). Bendingargrunnurin contains inflections, lemmas and morphological data for faroese words and names. Most words are taken from wordlists that were made for The Spell Checker (Rættstavarin) (*Rættstavarin*, n.d.), that were taken from a faroese dictionary (Jóhan Hendrik W. Poulsen, 1998). The names are from the name list from The Faroese Language Council (Málráðið) (*Bendingarunnurin*, n.d.).

3.7 No Language Left Behind

The faroese No Language Left Behind (NLLB) dataset consists of multiple types of data primary billingual text (bitext), mined bitext and monolingual Text. For faroese, the datasets consist of primary (*NLLB - Faroese - Primary*, n.d.) and mined bitext (*NLLB - Faroese - Mined*, n.d.). The primary bitext corpora are publicly available parallel corpora from a variety of sources. The mined bitext corpora are retrieved by large-scale bitext mining. The mined data includes all the English-centric directions and a subset of non-English-centric directions. Only the faroese data is used in this study.

Chapter 4

Methods

This chapter describes the methods used in this project, including the categorization of error types, data processing, training, evaluation.

4.1 Categorizing Error types

The categorizing of errors is an essential part of the GEC process, as it allows for a more structured approach to error correction. The errors are categorized into three main categories: **Spelling**, **Grammar** and **Punctuation**. The **Spelling** category includes errors that are related to spelling, such as missing or added letters, swapped letters, and incorrect capitalization. The **Grammar** category includes errors that are related to grammar, such as incorrect verb forms, incorrect noun forms, and incorrect word order. The **Punctuation** category includes errors that are related to punctuation, such as missing or added punctuation marks, and incorrect punctuation marks. Each of these categories is further divided into subcategories, which are described in the following sections.

4.1.1 Spelling Errors

Spelling error types will have a name that starts with either **Spelling mistake** or Replacement Pattern (REP). In general **Spelling mistake** refers to a word that is misspelled, while REP refers to a pattern that is replaced with an error, for example **Rep "rsk" and "rk"** refers to the pattern "rsk" being replaced with "rk" and vice versa. Any REP where there is an **and** between the two words, refers to a two way relation, where both patterns can be replaced with each other, if there is an **with** between the two words, it refers to a one way relation, where the first pattern is replaced with the second pattern, but not vice versa. In addition to this, there are **Double consonant** and **Vowel confusion**. **Double consonant** refers to any mistake where a consonant should be doubled, but isn't or vice versa, while **Vowel confusion** refers to a mistake where a vowel is confused with another vowel, for example **"a" and "æ"**

or **“i” and “y”**. Other error types that don’t fall into the above categories, will have a name that describes the error type, such as **Capital initial letter**.

4.1.2 Grammar Errors

Grammar error types will often have a name that describes the word class that is being corrected, for example for an inflexion error, it will have the name of the wordclass the word belongs to, such as **Adjectives**, **Nouns** or **Verbs**. A broad error type like **Verbs** can also have a more specific error type, such as **Verb Tense Agreement** or **Verb Subject agreement**, this indicates that if there is an overlap where the error type can fall under either **Verbs** or **Verb Tense Agreement**, then the more specific error type will take priority be used. In addition to this, there are some error types where it says **added**, **missing**, **important** or **redundant**. **Added** and **Redundant** refer to a character or a word that is added to the sentence incorrectly or is redundant, while **Missing** refers to a character that is missing from the sentence, and **Important** refers to a word that is important for the sentence, but is missing. Other error types that dont fall into the above categories, will have a name that describes the error type, such as **Nomen Agensis**.

4.1.3 Punctuation Errors

Punctuation errors are either called the name of the punctuation mark that is being corrected, such as **Comma** or **Period**, or the are preceded by **Punctuation -** if they are less common, such as **Punctuation - exclaimation mark** or **Punctuation - missing space**. In the same way as with the grammar errors, an error type can have a more specific error type, such as **Comma - listing**, where the listing error will take priority over the general **Comma** error type.

4.2 Data Processing

This chapter describes the data processing steps involved in preparing the data for training the models. The data processing pipeline includes data collection, cleaning, preprocessing, and augmentation. Each step is essential for ensuring the quality and integrity of the data used for training the models. The following sections provide a detailed overview of the data processing pipeline used in this study.

4.2.1 Data Collection

All the training data was from various online repositories and websites. The training data is comprised of Faroese text from various sources, including articles from wikipedia, news websites, and research papers, as well as social media posts, legal documents, posts from government institutions and books. Most of the data is available online and can be accessed through the respective websites. The formats of the data vary depending on the source, and the data was collected in the form of text files, CSV, json, jsonl, html, xml and pdf, so some preprocessing is needed to get it in a uniform format.

4.2.2 Data Cleaning & Preprocessing

The unlabeled data was cleaned using a mix of heuristics. To remove a lot of the foreign sentences, a blacklist of foreign characters was used to filter out sentences that contained these characters. Additionally a list of common danish and english words were used to remove foreign sentences. To get rid of some metadata, words and abbreviations like **img src**, **aspx**, **pid**, **newsid**, **html**, **date** were used to remove the sentences they occur in. Due to encoding errors in the data, a lot of the data was wrongly converted to ascii, but a lot of it could be reverse engineered by manually inspecting the data, and from context, get a mapping from the wrong encoding to the correct faroese character. For example the character **ð** was written as **Ã°** and the character **á** was written as **Ã¡** and so on. The **ð** character is also sometimes written as **đ**, so to make the dataset more uniform, it is converted to **ð**, which is the only one of them that can be written with a faroese keyboard without modifiers. The data was also cleaned by removing any html tags, and any other non-faroese characters. Some of the data has really long sentences, some of them up to 800.000 characters long, so they were split on the period character, excluding periods from abbreviations. All duplicate sentences were removed from the dataset. The dataset was shuffled to make sure the model doesn't overfit on the order of the data. The unlabeled dataset was saved as JavaScript Object Notation (JSON) for pretraining of the mt5 model and as a txt file for the corruption process.

UD Data

The labeled data from the Faroese UD repositories required minimal preprocessing. All annotated sentences were collected into a single JSON file to facilitate easier inspection and further processing. Duplicate sentences were then removed from the dataset. This JSON file was subsequently converted into the spaCy training format, a binary format consisting of a list of documents, where each document includes one or more labeled sentences. The dataset was then split into training and validation sets, using a 95/5 split.

Testset from MTD

The original corpus is structured in XML format, with each sentence enclosed in its own XML tag. Within each sentence, every word and punctuation mark is also wrapped in individual tags. When a word or punctuation mark is corrected, it is replaced by a revision tag that contains both the original and the corrected version of the text.

Data Snippet 1: An example of how a sentence in the private corpus is formatted (this is not a real sentence from the corpus, just a minimal example)

```
<s n="1">
    <w>Vit</w>
    <revision id="15">
        <original>
            <w>fóru</w>
        </original>
        <corrected>
            <w>fara</w>
        </corrected>
        <errors>
            <error xtype="past4pres" eid="0" />
        </errors>
    <revision id="16">
        <original>
            <w>niðaná</w>
        </original>
        <corrected>
            <w>niðan</w>
            <w>á</w>
        </corrected>
        <errors>
            <error xtype="adv4adv-prep" eid="0" />
        </errors>
    </revision>
    <w>fjallið</w>
    <w>í</w>
    <w>morgin</w>
    <c>.</c>
</s>
```

For each error in a sentence a pair of incorrect and correct sentences are generated. The incorrect sentence contains the error and the correct sentence is the same as the incorrect sentence, but with the error corrected. Figure 4 shows the sentences that are generated from the sentence in Data Snippet 1.

Vit $\left(\frac{\text{fóru}}{\text{fara}}\right)$ $\left(\frac{\text{niðaná}}{\text{niðan á}}\right)$ fjallið í morgin.

⬇

Vit fóru niðan á fjallið í morgin. Vit fara niðan á, fjallið í morgin.

Vit fara niðaná fjallið í morgin. Vit fara niðan á fjallið í morgin.

Figure 4: Example of a sentence with an error and the corresponding correct sentence, Translation: "We went up the mountain tomorrow. We are going up the mountain tomorrow."

Since the original sentences were written by students, a single word would sometimes contain multiple errors, so by manually inspecting the data, the errors were split into multiple sentences once again. An example of this is the sentence "Teir ætlaðu sær til

eina ærda bygd, men blivu næstan vilstir uppi á fjøllunum, tí mjørkin kom." Here the word **ærda** contains two spelling mistakes, the **æ** should be an **a** and the **d** should be an **ð**. Another effect from the original sentences being written by students, is that they somtimes write very long running sentences, so the sentences were cut off in a way that preserved the context of the error, but removes the redundant parts of the sentence. An example of this, is the sentence:

Vit hoyra í byrjanini í stuttsøguni, at Maria vaknar, og tað fyrsta, hon hugsar, er: "Hvar er telefonin?" og so brúkar hon heilar 36 min. á telefonini, áðrenn hon vakir mannin Súna og sigur: "Klokkan er farin av sjey, vit hava forsovið okkum, kom upp!"

The error here is that **og** should be capitalized because of the preceding **?**, so it can be reduced to:

Tað fyrsta, hon hugsar, er: "Hvar er telefonin?" og so brúkar hon heilar 36 min. á telefonini.

This is done to make the sentences more concise and reduce the noise that comes from having overly long sentences.

4.2.3 Corruption Process

Before delving into the specific methodologies employed, it is useful to understand the evolution of the error generation process, as it directly shaped several aspects of the system design. Initially, the grammatical error generation relied on simple word confusions, followed by a more systematic approach using inflection data from (*Sprotin*, n.d.) to introduce morphological errors. However, at this early stage, identifying the correct word class for each word was only possible by consulting dictionary entries. As a result, only unambiguous entries, those that occurred with a single word class, could be reliably processed. Words with multiple potential classes, depending on context, were uniformly treated as inflection errors, because of this, a noun could get an adjective's inflexion, introducing significant noise into the dataset.
This limitation highlighted the need for a POS tagger to disambiguate words based on context. As development progressed, a spaCy pipeline was trained, allowing each word to be tagged with its appropriate word class. The addition of a morphologizer further enriched the pipeline by providing access to detailed morphological features, enabling more precise and contextually appropriate inflectional errors.
Nonetheless, the inflection data from (*Sprotin*, n.d.) proved insufficient, particularly for verbs. To address this, a more comprehensive inflectional table was constructed using data from (*Bendingar*, n.d.) offering complete morphological paradigms and proper noun inflections. With this improved dataset, the morphologizer could be fully leveraged to guide and refine the error generation process, resulting in higher-quality synthetic data and more realistic error patterns.

Building on these earlier stages, the current corruption process is tightly integrated with the spaCy framework and makes extensive use of spaCy DocBin (DocBin) objects to store tagged linguistic data. At present, Faroese data is annotated with POS tags and morphological features provided by the morphologizer. However, the design is modular and extensible, allowing for the easy inclusion of additional annotations, such as dependency relations, once the corresponding components become reliable.
One major advantage of this setup is that serialized DocBins allow the tagging step to be performed only once. After tagging, errors can be introduced without the need to reprocess the original data. For each line in the original dataset, a corresponding list of valid corruption options is generated and stored in a JSON file. The structure is kept simple: line n in the original file maps directly to line n in the JSON file, ensuring straightforward alignment and minimizing complexity.
To generate the actual training data, this JSON file is paired with a distribution file. The distribution file defines the desired frequency and types of errors to be sampled, the dataset splits (training/validation), the unchanged percentage and which source datasets should be included. For spelling errors, a source file that already contains grammar mistakes is used. This file is then passed through the corruption process to generate the training data. This ensures that the spelling model is trained specifically to correct spelling errors, in an environment similar to where it will ultimately be applied.

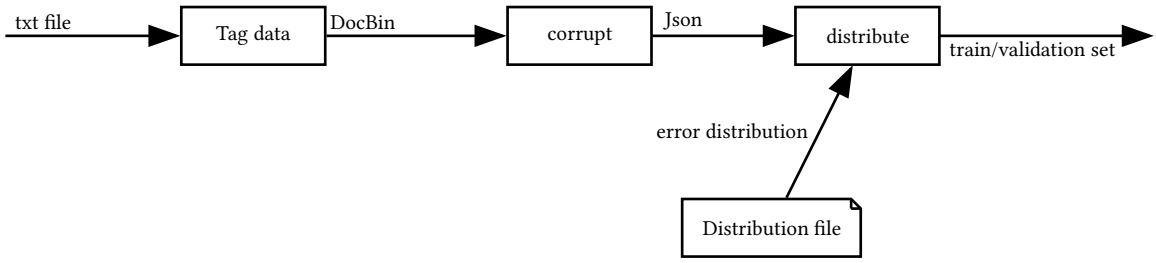


Figure 5: An overview of the corruption process used for data augmentation.

The advantage of this approach is that each step can be performed independently, allowing for easy updates and modifications to the corruption process without having to reprocess the entire dataset.

4.3 Training

This section describes the training process for the models used in this project. It includes details on the pretraining and fine-tuning of the mT5 model and the training of the spaCy pipeline components. All the mT5 models were trained on Nvidia RTX A6000 GPUs with 48GB of memory, and the spaCy pipeline was trained on a single Nvidia Geforce RTX 2080 Ti with 11GB of memory.

4.3.1 Pre-training

The pre-training of the mT5 model was conducted using the NanoT5 framework (Piotr Nawrot, 2023), a lightweight and flexible training infrastructure optimized for training T5-based models at scale. The model was trained from scratch on Faroese using a span

corruption objective, similar to that introduced in the original T5 model, where spans of text are masked and the model is trained to generate the missing content. This objective allows the model to learn both syntactic and semantic representations in a self-supervised manner.
Due to the lack of existing Faroese language models, the weights were randomly initialized. The training for the custom mT5 model was performed with the following hyperparameters:

- **Model dimensionality**: 384
- **Feed-forward dimensionality**: 1024
- **Key Value dimensionality**: 64
- **Encoder layers**: 6
- **Decoder layers**: 6
- **Attention heads**: 2
- **Input length**: 512
- **Masked language modeling probability**: 0.15
- **Mean noise span length**: 3.0
- **Optimizer**: AdamWScale
- **Learning rate schedule**: cosine decay
- **Initial learning rate**: $1e^{-04}$
- **Gradient clipping**: 1.0
- **Warmup steps**: 1000
- **Batch size**: 128
- **Accumulation**: 2
- **Epoch**: 1

The training for the mT5-base model was carried out using the following hyperparameters:

- **Model**: mT5-base
- **Precision**: bf16
- **Input length**: 512
- **Masked language modeling probability**: 0.15
- **Mean noise span length**: 3.0
- **Optimizer**: AdamWScale
- **Learning rate schedule**: cosine decay
- **Initial learning rate**: $1e^{-04}$
- **Gradient clipping**: 1.0
- **Warmup steps**: 1000
- **Batch size**: 64
- **Accumulation**: 4
- **Epoch**: 1

The training corpus consisted of approximately 5.2 million Faroese sentences, collected from various web and document sources. However, due to an oversight during preprocessing, duplicate sentences were not removed, which may have introduced mild data redundancy and impacted the model's exposure to diverse linguistic patterns.

A 95/5 training/validation split was used, resulting in a validation set of roughly 270,000 sentences. The model was evaluated against this set at the end of training. In addition, a checkpoint was saved every 10,000 steps to allow for resumption or retrospective model analysis.

4.3.2 Fine-tuning

The fine-tuning process was conducted using Ordbogen's custom training framework. The models were trained not only to generate corrected versions of erroneous input text, but also to output the associated error types for each correction. This output format aligns with recent developments in explainable GEC models, where corrections are annotated with specific categories for improved interpretability and evaluation.
An example of the model's input-output format is shown in Figure 6:

Vit gloyma ofta at nýtur vøkru tíð.

⬇

S Vit gloyma ofta at njóta hesa vøkru tíð. $A 6 8|||1601|||njóta $A 8 8|||1807|||hesa

Figure 6: Example of an input and output sentence. Translation: "We often forget to enjoy this beautiful time."

In the example on Figure 6, the corrected sentence is annotated with error type identifiers. Here, **1601** corresponds to a **Verbs** error, and **1807** represents a **Subject - missing error**. This format not only guides the model in generating accurate corrections but also provides structured error annotations that can be evaluated independently or used in downstream applications, such as automated feedback tools for language learners. By treating grammatical error correction as a conditional generation problem, this approach leverages the pre-trained capabilities of mT5 to adapt to Faroese.

The following hyperparameters were used during fine-tuning:

- **Learning rate**: $1e^{-05}$
- **Optimizer**: AdamW
- **Batch size**: 16
- **Accumulation**: 8
- **Warm-up steps**: 10000
- **Max sequence length**: 128
- **Max generation length**: 256

In order to monitor performance during training, the model was evaluated on a validation set at the end of each epoch. If the validation loss improved, a checkpoint of the model's weights was saved. This ensured that the best-performing model was always preserved, regardless of fluctuations in performance during subsequent epochs. In addition to these conditional checkpoints, complete model snapshots were saved every five epochs, regardless of validation loss. These periodic checkpoints were stored separately from the best-performing checkpoint to support retrospective analysis or to enable rollback to earlier stages of training if necessary.

To safeguard against unexpected interruptions, the training framework was also config-

ured to save a checkpoint in the event of a manual stop signal or system interruption. Furthermore, if an exception occurred during training, such as a CUDA out-of-memory error, a final "exit" checkpoint was saved automatically. This precaution allowed training to resume from the point of failure or to assist in debugging problematic behavior without losing significant training progress.
This evaluation and checkpointing strategy contributed to the robustness of the fine-tuning process, providing multiple recovery options while also ensuring consistent tracking of the model's best performance.

4.3.3 spaCy POS Tagger and Morphologizer

The POS tagger and morphologizer were trained using a spaCy pipeline built around a transformer architecture, which served as a shared encoder for all downstream linguistic components. Specifically, the pipeline leveraged the mBERT-cased model as a shared transformer backbone. This choice enabled the model to benefit from multilingual knowledge transfer while learning contextual embeddings adapted to Faroese through fine-tuning. The training was carried out using the following hyperparameters:

- **Model**: mBERT-cased
- **Optimizer**: AdamW
- **Weight decay**: 0.01
- **Gradient clipping**: 1.0
- **Initial learning rate**: $5e^{\{-05\}}$
- **Batch size**: 128
- **Accumulation**: 3

Due to the fixed-length limitations of BERT-based models, input sequences longer than the model's maximum window size must be split into manageable chunks. In this configuration, strided spans were used to produce overlapping token windows. Each window covered 128 tokens, with a stride of 96, meaning that each new window started 96 tokens after the previous one. This overlap preserved context across chunks and mitigated edge effects that might reduce tagging accuracy near window boundaries.
To aggregate the output of the transformer into a representation usable by downstream components, the pipeline employed a reduce mean pooling strategy. This pooling method calculates the mean of all token embeddings in a given span, producing a single fixed-size vector that summarizes the semantic content of the span.
The POS tagger and morphologizer were trained jointly on top of the shared transformer encoder. Both components used the TransformerListener architecture to extract contextualized token features from the mBERT model. This setup enabled multi-task learning, where the transformer encoder was fine-tuned for multiple linguistic objectives simultaneously. This approach is especially beneficial in low-resource settings, as shared training can lead to improved generalization across tasks.

Evaluation was performed every 200 training steps. To balance the evaluation between components, a weighted scoring system was used. Since the morphologizer includes two sub-components (POS tags and morphological features), their weights were split evenly, resulting in the following evaluation weights:

- **Tag accuracy**: 0.5
- **POS accuracy**: 0.25
- **Morphology accuracy**: 0.25

Custom label sets were used for both the POS tagger and the morphologizer. These labels were derived from the Faroese Universal Dependencies treebanks (*Faroese-Farpahc Treebank*, n.d.; *Faroese-OFT Treebank*, n.d.), ensuring alignment with standardized linguistic annotations.

4.4 Evaluation

This section describes the methods used to evaluate the performance of the models. The evaluation process includes the selection of appropriate metrics and the design of experiments and the rationale behind the choices. The evaluation methods are essential for assessing the effectiveness of the models and their ability to generalize to unseen data. Normally for a language like english, there would be multiple testsets publicly available on the internet, and many other studies would have used it to evaluate their models, so you would have other models to use as benchmarks, to compare the performance of your model, but for faroese, there are no such testsets available publicly. While there are some testsets available from Scandeval, they are not quite what is needed for this study, as they are for sentiment classification, Named Entity Recognition (NER), linguistic acceptability and reading comprehension.

The private corpus mentioned in Section 3.3, was being developed at the start this thesis, so it was not available for the first six months. Before the private corpus was available, no testset was available, so one was made. Using articles from (*Sprotin*, n.d.), and adding various errors to the sentences, a testset was created. The testset was then used to evaluate the models. The testset was not used for training or validation of the models, so it is a good representation of how well the models perform on unseen data.

4.4.1 Grammar & Spelling

When evaluating the grammar and spelling models, the precision, recall and $F_{0.5}$ scores were used as the evaluation metrics. For each errortype, the average of the scores was calculated. Then an overall score was calculated by taking the average of the scores for all errortypes. In the current state of the testset, error type does not have a comparable number of tests, so the average of the scores is not an ideal representation of the performance of the models. Another issue that is less apparent, is that each error type is not nessecarily equally important, for example, a category like **noun** inflexions is more important that something like **punctuation - exclaimation mark**, as **noun** inflexions are much more common than errors with exclaimation marks. Currently the testsets do not have a way to represent this, the way the models have been evaluated, has been a lot of manual inspection of the results, and looking at the wrong predictions, and seeing if the are actually wrong. Unfortunately, evaluating if a sentence is correct or

not, is not always as simple as have a single incorrect sentence, that is always corrected, to a single correct sentence. For example, with the sentence "Hann rakst úti á myrkt havi." The original "intended" correction was "Hann rakst úti á myrkum havi." since that was the original sentence, but a correction that the model gave was "Hann rakst út á myrkt hav." While this is not the intended correction, it is still a valid correction, and the model should not be penalized for this. So the correction is added to the testset as an alternative correction. This happened with a lot of the sentences, and goes to show that making a testset for a grammar model is not as simple as just having a sentence with a single error, and a single correction, since there are a lot of possible ways to get a correct sentence. The way the models will be evaluated is a combination of scores and manual inspection of the results, highlighting the most common and important error types, and also highlighting the shortcomings of the models.

Calculating Scores

The scores for the grammar and spelling models were calculated using precision, recall and F_β scores. Two ways of calculating the scores were considered, the first was to compare each word in the sentence, and calculate the TP, FP, TN and FN scores on a word level. The advantage of this method is that it is more granular, and has the capacity to give partial rewards for cases where the model was able to correct the incorrect word, but has introduced a new error. The trouble with this is, if an error is introduced, that changes the offset of the words in a sentence, for example by inserting a word, it can cause every following word to be misaligned, and the model will get a lot of FPs. The second method is to compare the whole sentence, and use a more coarse grained approach, where the model is only evaluated on whether it was able to correct the sentence or not. The advantage of this method is that it is more robust to changes in the sentence, and does not suffer from the misalignment issue. The disadvantage is that it is less granular, and does not give partial rewards for cases where the model was able to correct the incorrect word, but has introduced a new error. The first method was selected for this study, despite the risk of misalignment, it is rare enough to not be a major issue, and the upside of a finer grained score outweighed the risk.

4.4.2 spaCy pipeline

The evaluation of the spaCy pipeline was done using the validation set. This was done because the scarcity of the data made it impossible to have a seperate testset, as the training set was already very small, so every little bit of data was needed for training. The training set was split into a training and validation set, using a 95/5 split. For the overall evaluation of the pipeline, the accuracy of the POS tagger and morphologizer was used. And for each of the features of the morphologizer, the precision, recall and F_1 scores were used as the evaluation metrics. The results of the evaluation are presented in Table 1 and Table 2.

Chapter 5

Results

This chapter presents the results of the experiments conducted in this study. The results are organized into sections: Each section provides a detailed analysis of the performance of the models.

5.1 spaCy Pipeline

A spaCy pipeline was trained on the Faroese dataset using the following components: a POS tagger, morphologizer, lemmatizer, and dependency parser. The performance of each component was evaluated using either accuracy or precision, recall, and F_1 score, depending on what metric is relevant for the given component. The POS tagger and morphologizer were trained on the same dataset, while the lemmatizer and dependency parser were each trained on separate datasets. Due to limited data, the same training set could not be used for all components, the amounts of data available for each component is mentioned in their section. The results of the experiments are presented in the following subsections. To have a baseline, The Stanza model from stanford was used as a reference, since it also used data from UD, although it only uses (*Faroese-Farpahc Treebank*, n.d.). All metrics for Stanza are taken from the Stanza websites performance page (*Stanza Performance Page*, n.d.). The models are not using the same testset, so the results are not directly comparable, the comparisons are just to give an idea of how the models perform compared to each other.

5.1.1 POS Tagger & Morphologizer

The POS tagger and morphologizer were trained on the same dataset, greater detail about the dataset can be seen in Section 3.5.

On Table 1, is a comparison of accuracy between the spaCy POS tagger and morphologizer, trained for this thesis and the faroese Stanza model. For the sake of clarity, the metrics on Table 1, will be explained: The “Tag” refers to the fine-grained POS tag that is predicted by the POS tagger. The POS refers to the coarse-grained POS

tag, that is predicted by the morphologizer. The MORPH refers to the morphological features that are predicted by the morphologizer.

Model	Tag	POS	MORPH
spaCy	**93.39**	**97.80**	**94.03**
Stanza	91.56	96.51	92.92

Table 1: POS Tagger and MORPH accuracy comparison between the spaCy and Stanza models

While the results look good, after some manual inspection of some prediction, it is clear that the POS tagger struggles with words like **skrivað** (written) if the context is "Bókin var **skrivað** í ár." (The book was written this year.), **skrivað** is an adjective, but if the context is "Eg havi **skrivað** bókina í ár." (I have written the book this year.), then it is a verb. The model struggles to disambiguate these two cases, and it is not the only case where this happens, but it is the most common case. This is likely a problem with the training data, where there is not enough examples of the different contexts, and the model is not able to learn the difference. Just looking at the results, the spaCy model outperforms the Stanza model, when it comes to overall accuracy of the POS tagger and morphologizer, but with the limited data and both training sets having major limitations its not clear how their performance generalizes to unseen data. On Table 2, a more detailed view of the performance of the morphologizer is shown, the table shows the precision, recall and F_1 score for each feature.

Feat	Precision	Recall	F1
Case	96.06	95.54	95.80
Gender	95.32	95.28	95.30
NameType	98.48	97.01	97.74
Number	96.98	96.64	96.81
Definite	97.90	98.00	97.95
Mood	98.31	97.85	98.08
Person	96.19	95.77	95.98
Tense	97.60	96.70	97.15
VerbForm	96.75	95.14	95.94
PronType	95.67	94.91	95.29
Degree	93.25	93.84	93.54
Voice	97.22	92.11	94.59
NumType	98.06	94.39	96.19
Abbr	92.86	96.30	94.55
Foreign	96.23	92.73	94.44

Table 2: Morphologizer performance metrics per feature for the spaCy model

Looking at the results, the morphologizer performs reasonably well, on most features, especially on **Mood**, **Definite**, **Tense** and **NameType**. The **Mood** and **NameType** features are not used for anything in the pipeline, as of now, but its still good to

have them for future use. The **Definite** and **Tense** features are very important for making inflexion errors, so it is good that they are performing well. On the other hand, **Degree** is performing pretty poorly with a f1 score of 93.54%, which is unfortunate, since it is needed to make degree inflexion errors, which is hard as it is, and will be further complicated with the poor performance of the morphologizer. Two other features with poor performance are **Abbr** and **Foreign**, thankfully these features are not very important, since foreign words are not of interest, and abbreviations are are not handled by the models. **Voice** has a low recall, but a high precision, but since **Voice** is not used for any errors as of now, it is not a problem. The rest of the features have an acceptable performance. Overall the morphologizer performs well, for most of the features, that are important.

5.1.2 Lemmatizer & Dependency Parser

An experiment was conducted to see what performance the lemmatizer and dependeny parcer could achieve on the very limited data. On Table 3, the comparison between the lemmatizer and dependency parser performance metrics for the spaCy and Stanza models is shown.

Model	Lemma	UAS	LAS
spaCy	81.22	82.39	63.23
Stanza	**99.64**	**84.39**	**80.07**

Table 3: Dependency parser performance metrics for the spaCy model

Looking at the results, it is clear that the lemmatizer and dependency parser performance metrics for the spaCy model are not as good as the Stanza model. With the accuracy of the lemmatizer being 81.22% around every fifth lemma is incorrect, making it unusable. Looking at the dependency parcer, the Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) scores are 82.39% and 63.23% respectively. This means that the dependency parcer is not too far off the Stanza model, when it comes to detecting structural dependencies, performs very badly when predicting what grammatical roles the words play in the sentence. This result shows that the parser has a shallow understanding of the sentence structure. Table 4 goes more into detail about the performance of the dependency parser.

Feat	p	r	f
sents	87.22	90.59	88.87
cc	82.35	72.73	77.24
advmod	86.46	82.18	84.26
cop	73.17	28.04	40.54
nsubj	88.33	86.89	87.60
case	96.15	88.24	92.02
root	14.01	94.57	24.40
obl	68.09	66.32	67.19
mark	89.29	89.29	89.29
ccomp	50.00	60.00	54.55
obj	84.52	87.65	86.06
det	89.19	86.84	88.00
acl:relcl	75.68	70.00	72.73
conj	58.93	58.93	58.93
dep	50.00	10.04	16.72
nummod	100.00	88.89	94.12
advcl	68.18	71.43	69.77
nmod:poss	100.00	83.33	90.91
amod	90.00	81.82	85.71
vocative	100.00	33.33	50.00
appos	27.27	30.00	28.57
nmod	100.00	4.82	9.20
acl	62.50	62.50	62.50
aux	90.74	89.09	89.91
iobj	84.62	84.62	84.62
xcomp	66.67	40.00	50.00
compound:prt	77.27	77.27	77.27
nsubj:cop	0.00	0.00	0.00
flat:name	80.00	80.00	80.00
fixed	0.00	0.00	0.00
discourse	0.00	0.00	0.00

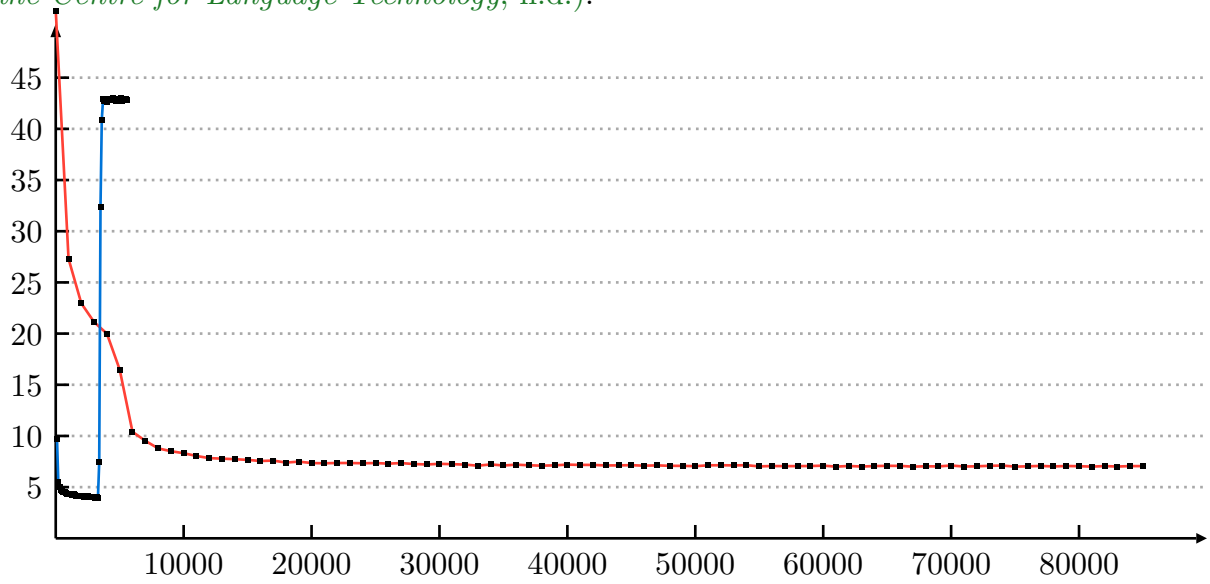
Table 4: Dependency parser performance metrics per type for the spaCy model

There are obviously some blank spots that the model has not learned at all, namely **nsubj:cop**, **fixed** and **discourse**. The **fixed** label occurs twice in the data, **nsubj:cop** occurs four times and **discourse** occurs 23 times, so it is not surprising that the model has not learned these labels. Other features that stand out are **root** and **nmod**. The **root** label has a pretty high recall of 94.57%, but with a precision of only 14.01% it is clear that the model has a hard time predicting the root of the sentence. The **nmod**

label has the opposite problem, with a precision of 100% and a recall of 4.82%, it is terrible at finding the **nmod** label, but when it does find it, it is correct, the label only occurs 66 times in the data, so it also is no surprise that it struggles to find it. In general, the models performance is all over the place, and with the small sample size, it is hard to say if these metrics generalize to the real world. Given the poor performance and lack of data, the lemmatizer and dependency parser were scrapped and not used in the pipeline.

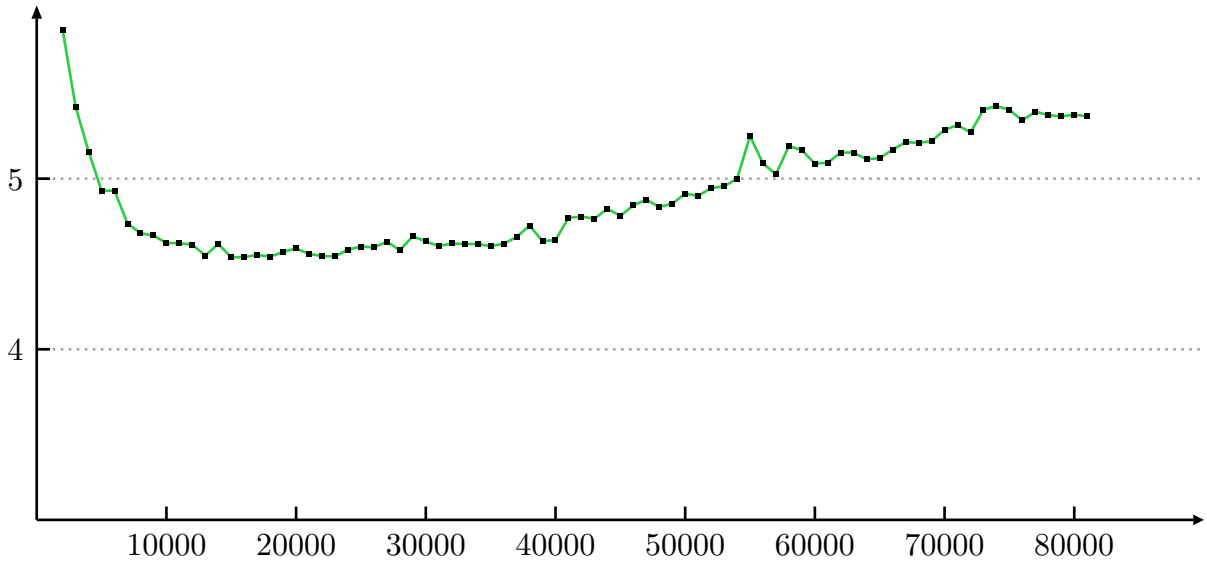
5.2 mT5 Pretraining

This section covers the experiments conducted to pretrain a mT5 model on faroese data. The initial results of the training were not promising, as most of the training destabilized after a few epochs. The dataset was a collection of wikipedia articles, blog posts and the Faroese corpora on Hugging Face and (*Faroese Language Resources by the Centre for Language Technology*, n.d.).



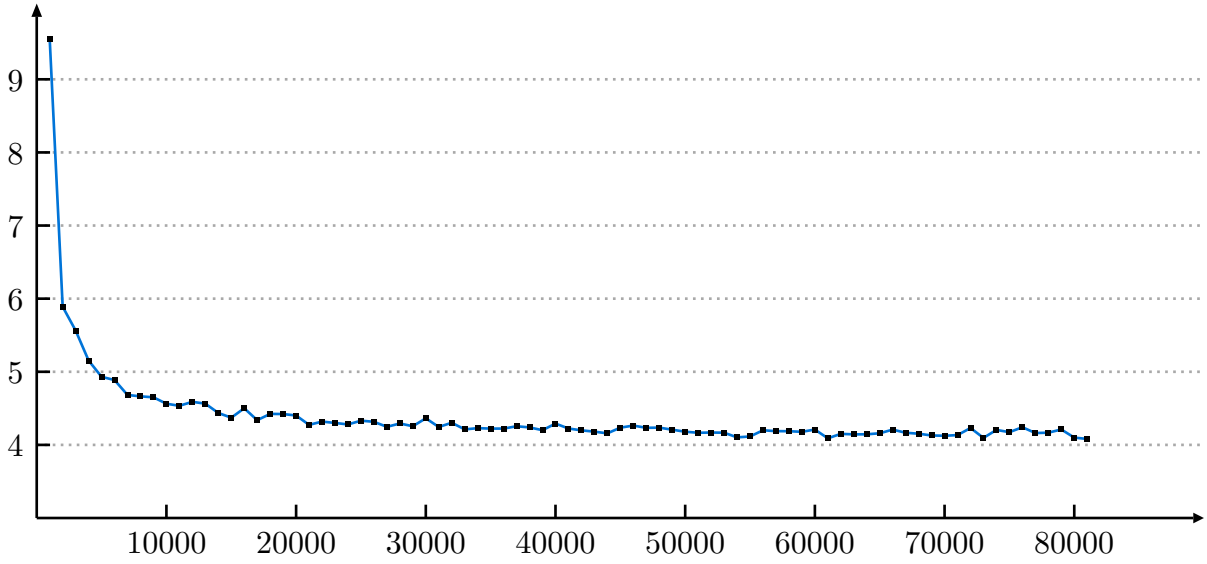
Graph 1: smaller mT5 pre-training loss, Red is Bitlinear and Blue is the regular smaller mT5

An initial hypothesis was that the data was not cleaned enough for the model to learn from, but looking at the graph for bitlinear, which didn't destabilize, it, This was possibly a part of the problem but not all of it. After more thorough cleaning, where all foreign characters were removed, and too short sentences were removed, as well as sentences that were too long. Additionally, common danish and english words, that do not appear in the faroese dictionary, were used to filter out sentences that were not in faroese. It was also at this point that there were some encoding errors in the data, which were fixed. After this, the training was restarted. and the result can be seen on Graph 2.



Graph 2: mT5 pre-training performance metrics

While the result is a bit better, it still destabilizes after 20,000 steps. At this point, the upsides and downsides of using a smaller model and bit linear were weighed. The smaller model would in theory be able to learn faster, but it seemed to be too small for the task, and while bitlinear stabilized the training, it was in the early stages of development, and had too many unknowns and could not be used in production, so in the end bitlinear was not used. Another pretraining was conducted using the larger mT5-base model. The training used the same dataset as the previous experiment.



Graph 3: mT5-base pre-training loss

This time the training was stable and the model was able to learn from the data. The training was stopped after one epoch, which is 81203 steps, but a checkpoint was extracted after 20,000 steps to work on finetuning. The loss plot can be seen on Graph 3.

5.3 mT5 Grammar Models

Two mT5-base models were trained for grammar correction. From a naming convention at Ordbogen, the models are named four letter names in the native language. The first model is named Kári (Kari internally because the danes on the team don't know how to pronounce accents), and the second model is named Bára (Bara internally for the same reason). The models were trained on different datasets and using different corruption methods. Further details about the datasets and corruption methods can be found in each of their sections, Section 5.3.1 and Section 5.3.2. The models were trained using the same training parameters. There are six error types that have testsets, but cant be made for now, because to make the errors in a manner that won't just introduce uncontrolled noise, a dependency parser is needed. These error types are **Verb Tense Inconsistent**, **Pronoun - Antecendent agreement**, **Verb - Subject agreement**, **Case - Verb**, **Case - Preposition** and **Gender**.

5.3.1 Kári

Kári was the first model trained, it was trained on 4.4M sentences, This dataset consisted of everything that was available at the time, this is everything on Hugging Face, MTDs website and and github repository, NLLB, Liebzig corpora and scraped articles from (*Faroe University Press*, n.d.). The dataset was cleaned as described in Section 4.2.2 and split into 95% training and 5% validation. Out of the 95%, 5% of the training data was un-corrupted to teach the model to not always correct. The full table with results can be seen in Section A.

On Table 5, the best performing error types, based on the $F_{0.25}$ score, are shown.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1465	Punctuation - exclamation mark	5	100%	0%	1	1	1	0%
1875	Rep "rsk" and "rk"	1	100%	0%	1	1	1	100%
1793	Numeral - inflexions	3	66.7%	0%	0.97	1	0.67	100%
1457	Numerals - large numbers	12	33.3%	0%	0.92	1	0.42	100%
1409	Spelling mistake - gi omlyd	19	57.9%	0%	0.89	0.92	0.58	81.8%
1415	Confusing "í" and "ið"	21	95.2%	0%	0.87	0.87	0.95	100%
1421	Spelling mistake - ki omlyd	14	92.9%	0%	0.87	0.87	0.93	100%

Table 5: Kári's top performing $F_{0.25}$ scores

When looking at the error types, where the model performed best, it's not as simple as the model performing best on the most common error types in the training data, but this is the case **Punctuation - exclaimation mark**, but not because exclamaition marks are all that common, but because an alternative correction for the errors is to use a comma instead of an exclaimation mark, and comma is one of the most represented error in the training data, this also means that the hit percentage for exclaimation marks is 0%. This highlights a significant problem with the testset, as the model is now getting max $F_{0.25}$ score for an error type that it didn't use at all, while it corrected the sentence in a correct manner, it gets scored as being good at exclaimation marks, which is not nessecarily the case, since the testset is not measuring it correctly in its current state. **Rep "rsk" and "rk"** is an example of the opposite, there are very few examples of this in the training data, with only 5800 examples, and additionally there is only one test sentence with the error type, so its hard to draw a conclusion about how this generalizes. **Numeral - inflexions** is another error type that has few test sentences, and the model has a high precision, but with so few sentences its not possible to say if this generalizes over a larger dataset. **Numerals - large numbers** has a perfect precision, but a low recall, this is because the model has learned to set the correct

"thousand" separator, but has not learned to correct swapped separators and decimal points, but this is a minor error overall and being careful to not change numbers too aggressively is not nessecarily a bad thing and large numbers that also have decimals are not common in the dataset. The two spelling mistake error types, **gi omlyd** and **ki omlyd** are fairly simple, but there are not that many examples of them, so the the low recall on **gi omlyd** is not a surprise and the high precision is good for so few examples in the training data. The last top performing error type **Confusing - "í" and "ið"** is on of the most common errors that people make. From the results it can be seen that the model makes mistakes on the test set, but the errors are on different words than the ones that are relevant to the error type, so the model is not actually making mistakes on the error type, itself but wrongly predicting other words in the sentence, which is a recurring problem.

Eth errors are the most common spelling / grammatical errors in faroese, and it is important that the model learns these. The errors are split into three types, swapped, missing, and added, they are called Eth - swapped ð (Eth - byttet ð), Eth - missing ð (Eth - mangler ð) and Eth - added ð (Eth - tilføjet ð) on Table 6.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
2003	Eth - byttet ð	10	20%	0%	0.59	0.67	0.2	50%
1393	Eth - mangler ð	120	45%	5.8%	0.79	0.83	0.45	0%
1395	Eth - tilføjet ð	108	29.6%	7.4%	0.65	0.7	0.3	84.4%

Table 6: Kári's Eth errors

Unfortunately, the performance is not as good as it could be, with the model only finding 20% of the Eth - byttet ð errors correcting them, the low precision comes from the model making mistakes on other words. In addition to this, the hit percentage is only 50%, which is not great, the model tagged one of the corrections as a Eth - tilføjet ð error, but since there are only two cases where it corrected the error, its hard to say if the model will mix the errors in general or if it's a standalone case. The model performs better on the Eth - mangler ð errors, with a precision of 0.83 and a recall of 45, this is a better result, but the model still misses half of the errors, which is not ideal and still makes too many errors, while it is one of the more complex errors, it is very common and well represented in the dataset. In addition to this, the hit percentage is 0%, this is due to an oversight and a bug in the corruption method, where the error type was not given priority over the inflexion types, so the model predicts inflexion errors instead of Eth - mangler ð, while this is not a grammatical error as such, it is definetly preferable to have the correct error type attached, so the end user can get a better and more precise description of the error that they are making. But ultimately the hit rate is easily fixed by giving Eth - mangler ð priority over inflexion errors, and

then re-corrupting the dataset, and continuing training for a few epochs. The last eth error, Eth - tilføjet ð, also has a far too low recall and precision, but at least the model has a better hit rate of 84.4%, but by inspecting the predictions, the model made, it is making mistakes on other words, and the hit percentage is actually better than the results show.

Inflexion errors are the second most common grammatical errors in faroese, so it is important that the model performs well on these. The main wordclasses that are relevant for this are **Adjectives**, **Nouns** and **Verbs**. The results for the inflexion errors for the three main wordclasses can be seen on Table 7.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1597	Adjectives	67	29.9%	6%	0.59	0.63	0.31	90%
1599	Nouns	99	34.3%	13.1%	0.63	0.67	0.32	88.2%
1601	Verbs	71	57.7%	8.5%	0.54	0.54	0.63	90.2%

Table 7: Kári's inflexion errors

Here it is quite clear that the model is underperforming, especially on **Nouns**, around a third of the predictions are wrong and on top of that, the recall is only 0.32. **Adjectives** and **Verbs** are performing better, when it comes to correctly predicting the errors, they still have lower precision, from wrong predictions on other words. The recall of **Verbs** is 0.63, which is not good, but its better than the other two wordclasses. A problem that stems from the POS tagger, is that it tends to mistake the wordclasses of certain **Adjectives** for **Verbs**, this happens for words like "skivað"(written), when the word is tagged with the wrong wordclass, it will then have the wrong inflexion errors made, so it is likely that a better POS tagger will improve performance on inflexions in general. When it comes to punctuation, the models performance is absolutely abysmal. It is so bad that it ends up making significantly more mistakes than it corrects, in fact most of its "corrections" are wrong, and here the problem of too poor quality training data couldn't be more clear. Looking at the error type **Comma - missing** it was only able to correct 0.6% of the errors, and it introduced errors on 23% of the sentences. The amount of comma errors in the training data was around 10%, so the amount of data was clearly not the issue here, it is simply a matter of what it learned from the data is not correct. Because of the fact that comma errors perform so badly, it introduces a lot of errors on other tests, and is one of the major reasons why the model performs so poorly in terms of precision. The biggest problem with Kári was that the dataset it was trained on was of too poor quality, and this put a limit on how well the model could learn. Another issue was the corruption method was too broad, and errors with gender and tense were made in cases where, from the context, its not possible to say if one is more grammatically correct than the other. An example of this is a sentence like "Hann er heima." (He is home.), an error could be introduced on "Hann" (He) where it's changed to "Hon" (She) or "er" (is) to "var" (was), errors like this will just confuse the model and encourage unnecessary correction, and this could be seen clearly through

manual inspection when running the testset. Additionally the model would lowercase names, which causes problems in a lot of testsets, and makes it difficult to evaluate if the model is actually bad at the error type being tested, or if its making wrong predictions because of the names being lowercased.

5.3.2 Bára

Bára was the second model that was trained, the corpora used for training was significantly smaller than the one used for Kári. It consisted of 1.5M sentences. After it became apparent that Kári hit a plateau, it was decided to train a second model with a more selective dataset. Anything that had questionable quality was removed, this means anything like wikipedia and scraped blog posts from unknown sources. The dataset combination of the following:

- two private datasets from MTD
- scraped papers from Fróðskaparrit and Fróðskapur (*Faroe University Press*, n.d.)
- scraped articles from Føroya Landssýri
- Sprotin parallel
- Faroese BLARK small
- Blog posts by Birgir Kruse
- Articles from Bróðrasamkoman
- A collection of books
- Articles from Kringvarp Føroya

These datasets are the ones used at the time of writing, and it is being looked into getting more data, so they may change in the future.
In addition to the smaller dataset, Bára not finetuned from the pretrained mT5 model, from Section 5.2, but the pretrained model on Hugging Face. Since the pretrained model from Section 5.2 was trained on an earlier version of the dataset Kári was trained on, and it was not good enough to be used for training, it was decided that it would be better to use the model on Hugging Face. Lastly, the corruption process was modified to be more selective when introducing errors, particularly when it comes to introducing gender errors and tense errors. This was one of the main problems with Kári, as the model ended up introducing a lot of errors by unnecessarily changing the gender and tense of words. At the time of writing, the model is still in the early stages of fine-tuning, so the results in this section are preliminary and will most likely change significantly in the future. The model has only trained for 47 epochs, compared to the 224 epochs that Kári has trained on a larger dataset, so the results are promising. The full table with results can be seen in Section B.

The top performing error types, for Bára, based on the $F_{0.25}$ score, with a perfect score of 1 are the following:

- Spelling mistake - “hesa/hesi” with “hesu”
- Punctuation - exclamation mark
- Numeral - inflexions

- Confusing “tí” and “ta”
- Rep “rsk” and “rk”
- Confusing “loyvi” and “loyvt”

The model is able to correct these errors with a high precision and recall, which is promising for the future, but it should still be noted that some of the training sets are quite small, and it is being worked on to get more data for these error types, so the results are subject to change.

Moving on to the more difficult error types to correct. The errors is something that the model is not very good at correcting.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
2003	Eth - byttet ð	10	60%	0%	0.84	0.86	0.67	100%
1393	Eth - mangler ð	119	40.3%	8.4%	0.4	0.39	0.45	16.7%
1395	Eth - tilføjet ð	109	31.2%	16.5%	0.55	0.57	0.38	52.9%

Table 8: Bára’s performance on Eth errors

The model outperforms Kári on Eth - byttet ð, with a higher precision and recall, but the model still struggles with Eth - mangler ð and Eth - tilføjet ð, both in terms of precision and recall, the hit rate on these two error types is also quite low, which is unfortunate, and needs improvement. When it comes to inflexion errors, it is also a mixed bag, the model performs well on some error types, but not so well on others as can be seen on Table 9.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1597	Adjectives	67	19.4%	4.5%	0.66	0.75	0.23	84.6%
1599	Nouns	99	23.2%	8.1%	0.55	0.57	0.33	65.2%
1601	Verbs	71	43.7%	11.3%	0.6	0.61	0.48	93.5%

Table 9: Bára’s inflexion errors

When it comes to **Adjectives** the model performs better than Kári, with a higher precision, but at the cost of a lower recall, while it would be nice to have a higher recall, the increased precision is more important. The same can be said for **Verbs**, the model has a higher precision, but a lower recall, not ideal, but an improvement. Unfortunately, for **Nouns** the model has a lower precision and recall, and the hit rate is low. Overall the inflexions have a low hit rate, which may mean that the model is still struggling with language understanding, and is not able to understand the context of the sentence

well enough to give the correct error type.

The model is still in the early stages of fine-tuning, so the results are promising, but there is still a lot of work to be done to get the model to perform well on all error types.

5.4 mT5 Spelling Model Rúna

The spelling model was trained on the same dataset as Kári, but the only focus was on spelling errors. The model was trained for 58 epochs. The poor quality of the dataset does not seem like it has affected the model as badly as Kári, as the model is able to correct spelling errors reasonably well. The training of the model was stopped fairly early due to the results not being promising at the time, but after a lot of work on the testset, and rerunning the tests, it turns out that the testset was not testing what it was supposed to test, and the model is actually able to correct spelling errors reasonably well. The error type ids that the model learned are now out of date, and this can be seen in the results, the hit rate on most error types is 0%, since the model is not used in production it is not a problem, as the model would need additional training regardless to be ready for production, and in the new training it would learn the new and correct error type ids.

On Table 10, the top performing error types, based on the $F_{0.25}$ score, are shown.

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1457	Numerals - large numbers	12	100%	0%	1	1	1	100%
1869	Rep "um$" and "un$"	3	100%	0%	1	1	1	0%
1875	Rep "rsk" and "rk"	1	100%	0%	1	1	1	0%
1409	Spelling mistake - gi omlyd	19	68.4%	0%	0.97	1	0.68	0%
1413	Spelling mistake - "hesa/ hesi" with "hesu"	7	57.1%	0%	0.96	1	0.57	0%
1421	Spelling mistake - ki omlyd	14	100%	0%	0.94	0.93	1	0%
1451	Capital initial letter	147	70.1%	0.7%	0.92	0.94	0.7	100%
1818	Spelling mistake - foreign words	3	33.3%	0%	0.89	1	0.33	0%
1870	Double consonant	27	40.7%	3.7%	0.85	0.92	0.41	0%
2091	Vowel confusion	25	36%	0%	0.83	0.9	0.38	0%
1780	Pronouns - reflexive	6	16.7%	0%	0.77	1	0.17	0%

Table 10: Rúna's top performing $F_{0.25}$ scores

Overall in the top cateories, the model performs really well, while the recall could certainly be improved, on around half of the error types, the precision is very high, so as far as preliminary results go, this is excellent. Looking at where the model performs well, its clear that the model is able to correct the simple spelling errors, but in addition to that, it is also able to correct **Numerals - large numbers**. At the bottom of Table 10,

is **Pronouns - reflexive**, which does not fit in with the spelling errors, but the error it was able correct was a missing d, which is a pattern for a spelling mistake, but in this case it results in going from an incorrect inflexion to a correct inflexion, so while the prediction is correct, it's not good that a grammar error is being corrected as a spelling error, as it will inform the user of a spelling error instead of the actual error. For most other error types, the precision falls drastically, and the recall is very low, for the most part, the low recall is a good thing, since it is only supposed to correct spelling errors, but the low precision is a problem, and is the reason that the model was not used in production. To get the model to perform better, it needs more training, it makes too many mistakes, on errors that dont have anything to do with spelling, and it is incapable of correcting errors like Eth - byttet ð, Eth - mangler ð or even the relatively simple **Accent** errors. While the spelling model is not as bad as Kári, when it comes to introducing errors, it is still not good enough to be used in production, and it needs more training.

Chapter 6

Conclusion

This thesis set out to explore the feasibility of developing foundational language models for ultra-low resource languages, using Faroese as a case study. In doing so, it addressed a pressing gap in NLP research: the scarcity of tools and annotated resources for languages with limited digital footprints. By focusing on GEC as a central task, and building supporting components such as a POS tagger, morphologizer, lemmatizer, and dependency parser, this work provides a grounded assessment of what is realistically achievable in such resource-constrained settings.

The results demonstrate that certain components, most notably the POS tagger and morphologizer trained using the spaCy framework, are viable with the currently available Faroese Universal Dependencies datasets. These tools deliver acceptable performance and offer a solid linguistic backbone for future grammatical modeling efforts.

However, other components present greater challenges. The initial GEC model faced significant limitations due to the quality of the training data and an overly simplistic corruption process, which prone to introducing errors rather than correcting them. Subsequent improvements to the corruption mechanism and the development of a second GEC model suggest more promising directions, although the model is still in early stages and requires further evaluation.

Conversely, both the lemmatizer and the dependency parser proved inadequate under current conditions. Their poor performance highlights a clear need for additional annotated data. These results reinforce the idea that, while transfer learning and lightweight architectures can help, some foundational level of annotated linguistic data remains essential.

Finally, the exploration of a spelling model offers a useful complement to GEC, addressing surface-level errors that frequently co-occur with deeper grammatical issues. Although not yet fully integrated, this line of work shows potential for improving overall correction quality in real-world scenarios.

In sum, this thesis illustrates both the opportunities and constraints involved in developing NLP systems for ultra-low resource languages. It provides practical insights into data needs, tool performance, and workflow design, contributions that can inform future efforts not only in Faroese, but across the broader landscape of digitally marginalized languages.

Chapter 7

Outlook

This section outlines the future directions and potential improvements for the Faroese NLP models and tools developed in this project. It discusses the challenges faced, the limitations of the current models, and the opportunities for further research and development.

7.1 Pretraining Faroese Models

Pretraining large language models remains a significant challenge for Faroese due to the substantial data requirements inherent in such models. High-quality pretraining not only demands a large volume of text, but also data that is linguistically diverse, well-formed, and representative of the language's real-world usage. Unfortunately, the dataset used in this study did not meet these standards, and its relatively low quality likely limited the effectiveness of the pretraining process.

Despite these constraints, the use of a standard mT5-base model, without Faroese-specific pretraining, yielded reasonably strong results in downstream tasks. This suggests that leveraging existing multilingual models can be a viable path forward, especially in low-resource settings. Looking ahead, exploring pre-trained models that are more closely aligned with Faroese, such as those trained on typologically similar Nordic or Germanic languages, could offer better transfer performance and reduce the dependency on large-scale Faroese corpora.

Ultimately, while fully training large models for Faroese from scratch may remain out of reach for now, strategic use of multilingual or regional models, coupled with targeted improvements in training data quality, holds promise for future progress.

7.2 Grammar Models

While the current GEC model for Faroese demonstrates promising performance, it remains limited in several key areas. One primary constraint lies in its token-based cor-

rection strategy, which restricts edits to surface-level replacements without considering deeper syntactic relationships. In contrast, dependency-based approaches, which utilize a syntactic parse of the sentence, could enable more structurally aware corrections, especially for complex grammatical constructions or long-distance agreement errors.

Further improvements in performance and linguistic accuracy could be achieved by incorporating richer grammatical analysis tools into the pipeline. At present, the supporting infrastructure for Faroese NLP is still relatively sparse. The current spaCy setup includes only a POS tagger and a morphologizer, both trained on top of a multilingual transformer. However, a full spaCy pipeline, including components like a dependency parser, named entity recognizer, lemmatizer, and sentence segmenter would offer a much more robust linguistic backbone for grammar correction.

Such components could provide the model with fine-grained linguistic signals that go beyond token classification. For example, a syntactic dependency tree could help the model distinguish between subjects and objects, improving its handling of agreement, case, and word order. Likewise, reliable sentence segmentation and lemmatization would enhance the model's ability to make consistent and contextually appropriate edits.

Looking ahead, the development of these tools, either by training them directly for Faroese or by adapting them through multilingual transfer learning, could significantly boost the quality and interpretability of grammar correction systems. Moreover, such infrastructure would not only benefit GEC, but also support a wide range of downstream applications.

7.3 Spelling Models

The current spelling correction model for Faroese shows encouraging early results, but it still faces notable challenges. One of the main limitations is that it tends to introduce too many false corrections, suggesting that the model is not yet precise enough in distinguishing between genuine errors and valid but less common word forms.

Another key issue lies in the evaluation process. The current test set contains too few actual spelling errors, which makes it difficult to reliably assess the model's effectiveness. This data imbalance skews evaluation metrics and may give a misleading impression of model performance. Without a test set that reflects the true frequency and variety of real-world spelling mistakes, it remains challenging to track progress accurately.

Looking forward constructing or curating a more representative evaluation dataset, either by introducing synthetic errors or collecting user-generated data, would allow for more meaningful performance assessment.

Importantly, an improved test set would not only provide a more accurate measure of

current performance but also help guide further development of the model. It would offer clearer insights into whether the model architecture is sufficient or if additional tools might be needed to better handle the complexities of Faroese spelling. At present, the limited nature of the test set makes it difficult to draw such conclusions with confidence.

7.4 POS Tagger and Morphologizer

The current POS tagging and morphological analysis models for Faroese have achieved strong performance, particularly given the low-resource nature of the language. Leveraging transformer-based architectures and fine-tuning on Faroese-specific data has allowed these models to perform competitively in core linguistic tasks. However, to reach the level of performance seen in high-resource languages, further improvements in training data quality and diversity are essential.

A significant limitation lies in the available annotated corpora. Much of the current training data, including resources such as (*Faroese-Farpahc Treebank*, n.d.), while valuable, is relatively outdated and may not reflect contemporary usage or spelling conventions. Moreover, one of the key annotated datasets used for training includes Wikipedia articles. While the annotations themselves are sound, the underlying texts are often not well written and can contain various grammatical or stylistic errors. This introduces noise into the training process and may negatively impact the model's ability to learn accurate grammatical patterns. Additionally, the reliance on Wikipedia content introduces a bias toward formal, third-person narrative structures, which further limits the model's ability to generalize to more diverse or conversational domains.

To move closer to the performance of models trained on high-resource languages, there is a need for the expansion and modernization of training and evaluation datasets. In particular, a new test set that better represents modern Faroese usage and reduces noise from issues in the source texts, such as poorly written or error-prone Wikipedia articles, would significantly improve both training outcomes and evaluation reliability. While the existing annotations are of high quality, the quality of the original texts can introduce inconsistencies that affect model learning. Additionally, diversifying the genre and register of the training data, by incorporating spoken language, informal writing, and dialog, would help ensure that models are robust across a broader range of real-world use cases.

Such improvements would not only enhance model accuracy but also support the development of downstream applications like grammar checkers, intelligent writing aids, and language learning tools tailored to Faroese speakers.

7.5 Lemmatizer

The development of a lemmatizer for Faroese remains an open challenge. In this study, the dataset used was too limited in size to produce a lemmatization model with reliable performance. However, the path forward appears promising. Unlike more complex

linguistic tasks, lemmatization generally does not require deep semantic understanding, making it more accessible for native speakers to contribute to the annotation process. The existing annotated datasets used for POS tagging and morphological analysis could serve as a strong foundation. Since these datasets already contain rich grammatical information, extending them with lemma annotations would be a relatively low-effort yet high-impact task. While conceptually straightforward, the process is time-consuming, which is why it was not undertaken as part of this thesis.
With community involvement or targeted annotation efforts, it should be feasible to significantly expand the lemmatization dataset and train a model that meets practical standards. By building on the infrastructure already in place and leveraging native speaker expertise, Faroese could achieve a robust lemmatization tool to support further NLP development in the language.

7.6 Dependency Parser

The development of a reliable dependency parser for Faroese remains a significant challenge. Current models demonstrate insufficient accuracy for practical use, largely due to the very limited size of the available training dataset. High-quality dependency parsing requires precise syntactic annotation, a task that demands advanced linguistic expertise, something that makes dataset creation particularly difficult in a low-resource language context.
One potential avenue for progress could be integrating the creation of a syntactically annotated corpus into academic projects. For instance, a master's thesis focused on Faroese dependency annotation could make a meaningful contribution. However, the number of Faroese master's students in relevant fields is extremely limited.
An alternative approach could involve engaging top-performing bachelor's students, either through supervised thesis projects or credited self-study modules. While bachelor students may require more hands-on supervision and guidance, this route may still be viable, especially with clear annotation guidelines and expert support. Over time, such initiatives could help gradually build the foundational resources needed to develop a robust Faroese dependency parser.

References

A Gold Standard Dependency Corpus for English. (2014). Association for Computational Linguistics. https://aclanthology.org/L14-1067/

Alla Rozovskaya, & Dan Roth. *Annotating ESL Errors Using Crowdsourcing*. Association for Computational Linguistics. https://aclanthology.org/W10-1004/

Bendingar. Retrieved February 19, 2025, from https://bendingar.fo/

Bendingarunnurin. Retrieved February 19, 2025, from https://bendingar.fo/um/

Chris Brockett, & William B. Dolan. *Correcting ESL Errors Using Phrasal SMT Techniques*. Association for Computational Linguistics. https://aclanthology.org/P06-1032

Christopher Bryant, Mariano Felice, & Ted Briscoe. *Automatic annotation and evaluation of error types for grammatical error correction*. Association for Computational Linguistics. https://aclanthology.org/P17-1074/

Claudia Leacock, Martin Chodorow, Michael Gamon, & Joel Tetreault. *Automated Grammatical Error Detection for Language Learners*. Morgan & Claypool Publishers. https://doi.org/10.1007/978-3-031-02153-4

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, & Peter J. Liu. (2019). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. https://doi.org/10.48550/arXiv.1910.10683

Daniel Dahlmeier, Hwee Tou Ng, & Siew Mei Wu. *Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English*. Association for Computational Linguistics. https://aclanthology.org/W13-1703

Daniel Jurafsky, & James H. Martin. (2021). *Speech and Language Processing* (3rd ed.). Prentice Hall. https://web.stanford.edu/~jurafsky/slp3/

Faroe University Press. Retrieved December 3, 2024, from https://ojs.setur.fo/

Faroese Language Resources by FoNLP. Retrieved December 3, 2024, from https://github.com/FoNLP/faroese_language_resources

Faroese Language Resources by The Centre for Language Technology. Retrieved December 3, 2024, from https://mtd.setur.fo/en/tilfeingi/swoof/product_cat-text/

Faroese Parsed Historical Corpus (FarPaHC). Retrieved February 19, 2025, from https://github.com/einarfs/farpahc

Faroese-FarPaHC Treebank. Retrieved November 29, 2024, from https://github.com/UniversalDependencies/UD_Faroese-FarPaHC

Faroese-OFT Treebank. Retrieved November 29, 2024, from https://github.com/UniversalDependencies/UD_Faroese-OFT

Ikumi Yamashita, Satoru Katsumata, Masahiro Kaneko, Aizhan Imankulova, & Mamoru Komachi. *Cross-lingual Transfer Learning for Grammatical Error Correction*. International Committee on Computational Linguistics. https://aclanthology.org/2020.coling-main.415/

Jeffery Lichtarge, Chris Alberti, Daniel Andor, Emily Pitler, & Z. Wei. *Corpora Generation for Grammatical Error Correction*. Association for Computational Linguistics. https://aclanthology.org/N19-1333/

Joakim Nivre. (2015). *Towards a Universal Grammar for Natural Language Processing*. https://doi.org/10.1007/978-3-319-18111-0_1

Jóhan Hendrik W. Poulsen. (1998). *Føroysk Orðabók*. Føroya Fróðskaparfelag.

Katrin Næs. (2005). *Færøsk retskrivning - resultater fra en undersøgelse af færøske stile*. http://ojs.statsbiblioteket.dk/index.php/sin/issue/archive

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, & Oleksandr Skurzhanskyi. *GECToR - Grammatical Error Correction: Tag, Not Rewrite*. Association for Computational Linguistics. https://aclanthology.org/2020.bea-1.16/

Li Guo, Keith Ross, Zifan Zhao, George Andriopoulos, Shuyang Ling, Yufeng Xu, & Zixuan Dong. *Cross Entropy versus Label Smoothing: A Neural Collapse Perspective*. Retrieved May 24, 2025, from https://arxiv.org/abs/2402.03979

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, & Colin Raffel. (2021). *mT5: A massively multilingual pre-trained text-to-text transformer*. https://doi.org/10.48550/arXiv.2010.11934

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Magdalena Guz, & Ondrej Bojar. *Approaching neural grammatical error correction as a low-resource machine translation task*. https://aclanthology.org/N18-1055/

Martin Chodorow, Claudia Leacock, & Joel R. Tetreault. *Detection of Grammatical Errors Involving Prepositions*. Association for Computational Linguistics. https://aclanthology.org/W07-1604/

Masahiro Kaneko, & Mamoru Komachi. *Encoder-Decoder Models Can Benefit from Pre-trained Masked Language Models in Grammatical Error Correction*. Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.391/

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, & Adriane Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*. https://doi.org/10.5281/zenodo.1212303

NLLB - Faroese - Mined. Retrieved March 4, 2025, from https://github.com/facebookresearch/fairseq/blob/nllb/examples/nllb/modeling/scripts/flores200/lang_pairs_mine.txt

NLLB - Faroese - Primary. Retrieved March 4, 2025, from https://github.com/facebookresearch/fairseq/blob/nllb/examples/nllb/modeling/scripts/flores200/lang_pairs_primary.txt

Penn Parsed Corpora of Historical English. Retrieved February 19, 2025, from https://www.ling.upenn.edu/hist-corpora/

Piotr Nawrot. (2023). *nanoT5: Fast & Simple Pre-training and Fine-tuning of T5 Models with Limited Resources*. Association for Computational Linguistics. https://doi.org/10.48550/arXiv.2309.02373

Reut Tsarfaty, Joakim Nivre, & Evelina Andersson. (2010). *Statistical Parsing of Morphologically Rich Languages: What, How and Whither*. https://aclanthology.org/W10-1401/

Roman Grundkiewicz, & Marcin Junczys-Dowmunt. *Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data*. Association for Computational Linguistics. https://aclanthology.org/W19-4427/

Ryan McDonald, Fernando Pereira, Kiril Ribarov, & Jan Hajič. (2005). *Non-Projective Dependency Parsing using Spanning Tree Algorithms*. https://aclanthology.org/H05-1066

Rættstavarin. Retrieved February 19, 2025, from https://divvun.org/

Schneiderkamp Lab Github Repository. Retrieved May 24, 2025, from https://github.com/schneiderkamplab/nanoT5

Siew Mei Ng, & Hwee Tou Ng. *The CoNLL-2014 Shared Task on Grammatical Error Correction*. Association for Computational Linguistics. https://aclanthology.org/W14-1701

Slav Petrov, Dipanjan Das, & Ryan McDonald. (2012). *A Universal Part-of-Speech Tagset*. Association for Computational Linguistics. https://aclanthology.org/L12-1115/

Sprotin. Retrieved December 3, 2024, from https://www.sprotin.fo/

Stanza Performance Page. Retrieved May 21, 2025, from https://stanfordnlp.github.io/stanza/performance.html

Timothy Dozat. (2017). *Arc-Factored Biaffine Dependency Parsing* [Stanford University]. https://nlp.stanford.edu/~manning/dissertations/tim-dozat-dissertation-augmented.pdf

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, & Yuji Matsumoto. *Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners*. Asian Federation of NLP. https://aclanthology.org/I11-1017

Trond Trosterud's tools for Faroese. Retrieved February 19, 2025, from https://gtweb.uit.no/cgi-bin/smi/smi.cgi?text=%C3%81+tunguni+eru+sm%C3%A1ar+tenn.&action=analyze&lang=fao&plang=eng

Trong Huy Phan, & Kazuma Yamamoto. *Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses*. Retrieved May 24, 2025, from https://arxiv.org/abs/2006.01413

UDConverter. Retrieved February 19, 2025, from https://github.com/thorunna/UDConverter

Universal Dependencies v2: An evergrowing multilingual treebank collection. (2020). Association for Computational Linguistics. https://aclanthology.org/2020.lrec-1.497/

Unsupervised Cross-lingual Representation Learning at Scale. (2020). Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.747

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. https://doi.org/10.48550/ARXIV.1706.03762

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, & Luke Zettlemoyer. *Multilingual Denoising Pre-training for Neural Machine Translation*. https://aclanthology.org/2020.tacl-1.47

Zheng Yuan, & Ted Briscoe. *Grammatical error correction using neural machine translation*. https://aclanthology.org/N16-1042/

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, & Andrew Ng. *Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction*. Association for Computational Linguistics. https://aclanthology.org/N18-1057/

APPENDICES

Appendix A

Kári

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
2073	Abbreviations	10	0%	10%	0	0	0	0%
1753	Accent	6	16.7%	50%	0.24	0.25	0.17	100%
1597	Adjectives	67	29.9%	6%	0.59	0.63	0.31	90%
1796	Adjectives - comparison	13	0%	46.2%	0.79	0.82	0.47	0%
1734	Adverb or adjective	6	0%	16.7%	0	0	0	0%
1381	Apostrophes	15	0%	0%	0	0	0	0%
1451	Capital initial letter	150	60.7%	8.7%	0.36	0.35	0.61	98.9%
1741	Case - preposition	94	8.5%	27.7%	0.15	0.16	0.08	0%
1742	Case - verb	48	8.3%	16.7%	0.27	0.31	0.09	0%
1425	Commas	9	0%	22.2%	0	0	0	0%
1842	Commas - apposition	2	0%	100%	0	0	0	0%
1744	Commas - conjunction	3	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1825	Commas - extra	174	23.6%	16.1%	0.25	0.26	0.19	100%
1839	Commas - listing	9	11.1%	22.2%	0.17	0.2	0.05	0%
1838	Commas - missing	783	0.6%	23.1%	0.02	0.02	0.01	60%
1819	Compound nonword	18	0%	11.1%	0	0	0	0%
1389	Compound words	170	17.6%	11.8%	0.71	0.85	0.19	93.3%
1431	Compound words - "-megin"	7	0%	0%	0	0	0	0%
1415	Confusing "í" and "ið"	21	95.2%	0%	0.87	0.87	0.95	100%
2071	Confusing "lítið" and "fá"	1	0%	0%	0	0	0	0%
1890	Confusing "loyvi" and "loyvt"	1	0%	0%	0	0	0	0%
1891	Confusing "meira" and "fleiri"	4	0%	0%	0	0	0	0%
1797	Confusing "tí" and "ta"	2	50%	0%	0.5	0.5	0.5	100%
1892	Confusing "vilja"	3	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
	and "hevði"							
1788	Conjunction	4	25%	0%	0.47	0.5	0.25	100%
1887	Conjunction - extra	13	0%	23.1%	0.45	0.75	0.06	0%
1773	Conjunction - missing	13	0%	7.7%	0	0	0	0%
1411	Dialog	6	0%	66.7%	0.12	0.13	0.1	0%
1870	Double consonant	27	3.7%	11.1%	0.19	0.25	0.04	0%
2003	Eth - byttet ð	10	20%	0%	0.59	0.67	0.2	50%
1393	Eth - mangler ð	120	45%	5.8%	0.79	0.83	0.45	0%
1395	Eth - tilføjet ð	108	29.6%	7.4%	0.65	0.7	0.3	84.4%
1762	Gender	36	38.9%	11.1%	0.59	0.61	0.4	0%
1385	Hyphen	17	5.9%	5.9%	0.23	0.5	0.02	100%
1810	Important "ið"	47	0%	27.7%	0	0	0	0%
1399	Name - inflexions	64	1.6%	42.2%	0.06	0.06	0.05	0%
1738	Nomen agensis	4	0%	0%	0	0	0	0%
1599	Nouns	99	34.3%	13.1%	0.63	0.67	0.32	88.2%
1791	Nouns - genetives	8	0%	12.5%	0	0	0	0%
1793	Numeral - inflexions	3	66.7%	0%	0.97	1	0.67	100%
1459	Numerals	23	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1457	Numerals - large numbers	12	33.3%	0%	0.92	1	0.42	100%
1641	Ordinal number	46	4.3%	37%	0.07	0.07	0.04	100%
1775	Particle - important	3	0%	0%	0	0	0	0%
1756	Particle - redundant	1	0%	0%	0	0	0	0%
1439	Period	200	0%	27.5%	0.06	0.08	0.01	0%
1801	Preposition	16	18.8%	12.5%	0.4	0.43	0.19	100%
1800	Preposition - missing	5	20%	40%	0.22	0.33	0.04	100%
1783	Pronoun - Antecedent Agreement	2	0%	0%	0	0	0	0%
1794	Pronouns	16	31.3%	6.3%	0.65	0.71	0.28	80%
1780	Pronouns - reflexive	6	0%	16.7%	0	0	0	0%
1453	Proper noun	103	23.3%	7.8%	0.51	0.54	0.25	0%
1820	Punctuation - abbreviation	2	0%	0%	0	0	0	0%
1821	Punctuation - colon	15	6.7%	20%	0.35	0.38	0.17	0%
1836	Punctuation - dialog	23	0%	13%	0	0	0	0%
1465	Punctuation	5	100%	0%	1	1	1	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
	- exclamation mark							
1854	Punctuation - missing space	9	0%	66.7%	0.42	0.5	0.11	0%
1845	Punctuation - question mark	10	0%	0%	0	0	0	0%
1823	Punctuation - semicolon	12	0%	16.7%	0	0	0	0%
1877	Rep "ks" with "x"	1	0%	0%	0	0	0	0%
1894	Rep "mann" with "man"	25	0%	0%	0	0	0	0%
1875	Rep "rsk" and "rk"	1	100%	0%	1	1	1	100%
1869	Rep "um$" and "un$"	3	33.3%	66.7%	0.35	0.33	1	100%
1861	Spelling - "vera" and "verða"	48	4.2%	22.9%	0.08	0.09	0.04	0%
1687	Spelling mistake	66	24.2%	7.6%	0.47	0.5	0.23	0%
1413	Spelling mistake - "hesa/hesi" with "hesu"	7	57.1%	0%	0.57	0.57	0.57	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1818	Spelling mistake - foreign words	3	0%	0%	0	0	0	0%
1409	Spelling mistake - gi omlyd	19	57.9%	0%	0.89	0.92	0.58	81.8%
1421	Spelling mistake - ki omlyd	14	92.9%	0%	0.87	0.87	0.93	100%
1807	Subject - missing	8	12.5%	12.5%	0.84	0.97	0.27	100%
1774	Verb - finit verbum	3	0%	0%	0.75	1	0.15	0%
1784	Verb - infinitive	2	0%	0%	0	0	0	0%
1736	Verb Subject agreement	20	20%	20%	0.28	0.29	0.21	0%
1811	Verb Tense Inconsistent	2	0%	50%	0	0	0	0%
1601	Verbs	71	57.7%	8.5%	0.54	0.54	0.63	90.2%
1633	Word confusion	9	55.6%	11.1%	0.62	0.63	0.56	100%
1808	Word order	2	0%	0%	0	0	0	0%

Appendix B

Bára

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
2073	Abbreviations	12	100%	0%	1	1	1	100%
1753	Accent	6	33.3%	16.7%	0.26	0.25	0.4	100%
1597	Adjectives	67	19.4%	4.5%	0.66	0.75	0.23	84.6%
1796	Adjectives - comparison	13	7.7%	23.1%	0.55	0.75	0.1	100%
1734	Adverb or adjective	6	66.7%	0%	0.97	1	0.67	100%
1381	Apostrophes	15	6.7%	13.3%	0.22	0.25	0.07	100%
1451	Capital initial letter	147	61.9%	12.2%	0.51	0.5	0.63	100%
1741	Case - preposition	94	9.6%	18.1%	0.13	0.12	0.19	0%
1742	Case - verb	48	12.5%	10.4%	0.08	0.08	0.12	0%
1425	Commas	9	22.2%	11.1%	0.3	0.67	0.03	0%
1842	Commas - apposition	2	50%	0%	0.94	1	0.5	0%
1744	Commas - conjunction	3	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1825	Commas - extra	174	20.7%	13.8%	0.18	0.18	0.17	97.2%
1839	Commas - listing	9	66.7%	11.1%	0.4	0.39	0.58	0%
1838	Commas - missing	783	26.8%	11.5%	0.34	0.35	0.27	99.5%
1819	Compound nonword	18	16.7%	11.1%	0.28	0.3	0.13	100%
1389	Compound words	60	38.3%	6.7%	0.81	0.84	0.47	100%
1431	Compound words - "-megin"	7	85.7%	0%	0.99	1	0.84	100%
2085	Compound words - with or without complement	108	0.9%	9.3%	0.43	0.68	0.06	100%
1415	Confusing "í" and "ið"	21	100%	0%	0.96	0.95	1	100%
2071	Confusing "lítið" and "fá"	1	0%	0%	0	0	0	0%
1890	Confusing "loyvi" and "loyvt"	1	100%	0%	1	1	1	100%
1891	Confusing "meira" and "fleiri"	4	25%	0%	0.85	1	0.25	100%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1797	Confusing "tí" and "ta"	2	100%	0%	1	1	1	100%
1892	Confusing "vilja" and "hevði"	3	0%	0%	0	0	0	0%
1788	Conjunction	4	25%	25%	0.47	0.5	0.25	100%
1887	Conjunction - extra	13	0%	7.7%	0	0	0	0%
1773	Conjunction - missing	13	0%	15.4%	0	0	0	0%
2093	Consonant confusion	5	0%	20%	0	0	0	0%
1411	Dialog	6	0%	50%	0.22	0.25	0.08	0%
1870	Double consonant	27	22.2%	3.7%	0.73	0.86	0.22	100%
2003	Eth - byttet ð	10	60%	0%	0.84	0.86	0.67	100%
1393	Eth - mangler ð	119	40.3%	8.4%	0.4	0.39	0.45	16.7%
1395	Eth - tilføjet ð	109	31.2%	16.5%	0.55	0.57	0.38	52.9%
1762	Gender	36	25%	2.8%	0.75	0.83	0.28	0%
1385	Hyphen	17	23.5%	5.9%	0.57	0.8	0.1	100%
1810	Important "ið"	47	6.4%	10.6%	0.39	0.8	0.04	100%
1399	Name - inflexions	64	1.6%	14.1%	0.07	0.09	0.02	0%
1738	Nomen agensis	4	0%	25%	0	0	0	0%
1599	Nouns	99	23.2%	8.1%	0.55	0.57	0.33	65.2%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1791	Nouns - genetives	8	0%	25%	0	0	0	0%
1793	Numeral - inflexions	3	100%	0%	1	1	1	100%
1459	Numerals	23	0%	13%	0.15	0.2	0.03	0%
1457	Numerals - large numbers	12	91.7%	8.3%	0.85	0.85	1	100%
1641	Ordinal number	46	87%	8.7%	0.64	0.63	0.88	100%
1775	Particle - important	3	33.3%	0%	0.9	1	0.34	0%
1756	Particle - redundant	1	0%	0%	0	0	0	0%
1439	Period	200	0.5%	18%	0.19	0.23	0.05	0%
1801	Preposition	16	12.5%	18.8%	0.08	0.08	0.13	100%
1800	Preposition - missing	5	20%	20%	0.35	0.34	0.36	100%
1783	Pronoun - Antecedent Agreement	2	0%	0%	0	0	0	0%
1794	Pronouns	14	0%	14.3%	0	0	0	0%
1780	Pronouns - reflexive	6	0%	33.3%	0	0	0	0%
1453	Proper noun	103	11.7%	6.8%	0.26	0.29	0.12	0%
1821	Punctuation - colon	15	6.7%	20%	0.35	0.38	0.18	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1836	Punctuation - dialog	23	0%	21.7%	0	0	0	0%
1465	Punctuation - exclamation mark	5	100%	0%	1	1	1	0%
1854	Punctuation - missing space	9	0%	33.3%	0	0	0	0%
1845	Punctuation - question mark	10	0%	10%	0	0	0	0%
1823	Punctuation - semicolon	12	0%	16.7%	0	0	0	0%
1877	Rep "ks" with "x"	1	0%	0%	0	0	0	0%
1894	Rep "mann" with "man"	25	0%	12%	0	0	0	0%
1875	Rep "rsk" and "rk"	1	100%	0%	1	1	1	100%
1869	Rep "um$" and "un$"	3	33.3%	0%	0.89	1	0.33	100%
1861	Spelling - "vera" and "verða"	48	54.2%	8.3%	0.45	0.45	0.55	88.5%
1687	Spelling mistake	66	7.6%	7.6%	0.26	0.31	0.07	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1413	Spelling mistake - "hesa/ hesi" with "hesu"	7	100%	0%	1	1	1	100%
1818	Spelling mistake - foreign words	3	0%	0%	0	0	0	0%
1409	Spelling mistake - gi omlyd	19	84.2%	0%	0.59	0.58	0.82	100%
1421	Spelling mistake - ki omlyd	14	92.9%	0%	0.87	0.87	0.93	100%
1807	Subject - missing	8	12.5%	12.5%	0.44	0.86	0.05	100%
1774	Verb - finit verbum	3	0%	0%	0	0	0	0%
1784	Verb - infinitive	2	0%	100%	0	0	0	0%
1736	Verb Subject agreement	20	25%	15%	0.57	0.63	0.25	0%
1811	Verb Tense Inconsistent	2	0%	50%	0	0	0	0%
1601	Verbs	71	43.7%	11.3%	0.6	0.61	0.48	93.5%
2091	Vowel confusion	25	40%	12%	0.61	0.63	0.42	100%
1633	Word confusion	9	44.4%	11.1%	0.65	0.67	0.44	100%
1808	Word order	2	0%	0%	0	0	0	0%

Appendix C

Kári

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1409	Spelling mistake - gi omlyd	19	68.4%	0%	0.97	1	0.68	0%
1413	Spelling mistake - "hesa/ hesi" with "hesu"	7	57.1%	0%	0.96	1	0.57	0%
1457	Numerals - large numbers	12	100%	0%	1	1	1	100%
1780	Pronouns - reflexive	6	16.7%	0%	0.77	1	0.17	0%
1818	Spelling mistake - foreign words	3	33.3%	0%	0.89	1	0.33	0%
1819	Compound nonword	18	16.7%	0%	0.54	1	0.06	0%
1839	Commas - listing	9	11.1%	0%	0.47	1	0.05	0%
1869	Rep "um$" and "un$"	3	100%	0%	1	1	1	0%
1875	Rep "rsk" and "rk"	1	100%	0%	1	1	1	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
2073	Abbreviations	12	100%	0%	1	1	1	100%
1451	Capital initial letter	147	70.1%	0.7%	0.92	0.94	0.7	100%
1421	Spelling mistake - ki omlyd	14	100%	0%	0.94	0.93	1	0%
1870	Double consonant	27	40.7%	3.7%	0.85	0.92	0.41	0%
2091	Vowel confusion	25	36%	0%	0.83	0.9	0.38	0%
1687	Spelling mistake	66	22.7%	4.5%	0.6	0.68	0.21	0%
1395	Eth - tilføjet ð	109	3.7%	1.8%	0.33	0.67	0.04	0%
1753	Accent	6	50%	16.7%	0.61	0.6	0.75	0%
1399	Name - inflexions	64	1.6%	1.6%	0.18	0.5	0.02	0%
1597	Adjectives	67	1.5%	1.5%	0.17	0.5	0.01	0%
1601	Verbs	71	2.8%	2.8%	0.25	0.5	0.03	0%
2003	Eth - byttet ð	10	10%	10%	0.41	0.5	0.11	0%
1453	Proper noun	103	1%	2.9%	0.09	0.2	0.01	0%
1741	Case - preposition	94	1.1%	0%	0.09	0.2	0.01	0%
1393	Eth - mangler ð	119	0.8%	1.7%	0.07	0.14	0.01	0%
1825	Commas - extra	174	0.6%	1.1%	0.06	0.14	0.01	0%
1838	Commas - missing	783	0.3%	2.3%	0.02	0.03	0	0%
1381	Apostrophes	15	0%	6.7%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1385	Hyphen	17	0%	5.9%	0	0	0	0%
1389	Compound words	60	0%	0%	0	0	0	0%
1411	Dialog	6	0%	0%	0	0	0	0%
1415	Confusing "í" and "ið"	21	0%	4.8%	0	0	0	0%
1425	Commas	9	0%	11.1%	0	0	0	0%
1431	Compound words - "-megin"	7	0%	0%	0	0	0	0%
1439	Period	200	0%	3.5%	0	0	0	0%
1459	Numerals	23	0%	8.7%	0	0	0	0%
1465	Punctuation - exclamation mark	5	0%	0%	0	0	0	0%
1599	Nouns	99	0%	0%	0	0	0	0%
1633	Word confusion	9	0%	0%	0	0	0	0%
1641	Ordinal number	46	0%	4.3%	0	0	0	0%
1734	Adverb or adjective	6	0%	0%	0	0	0	0%
1736	Verb Subject agreement	20	0%	5%	0	0	0	0%
1738	Nomen agensis	4	0%	0%	0	0	0	0%
1742	Case - verb	48	0%	2.1%	0	0	0	0%
1744	Commas - conjunction	3	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1756	Particle - redundant	1	0%	0%	0	0	0	0%
1762	Gender	36	0%	5.6%	0	0	0	0%
1773	Conjunction - missing	13	0%	0%	0	0	0	0%
1774	Verb - finit verbum	3	0%	0%	0	0	0	0%
1775	Particle - important	3	0%	0%	0	0	0	0%
1783	Pronoun - Antecedent Agreement	2	0%	0%	0	0	0	0%
1784	Verb - infinitive	2	0%	0%	0	0	0	0%
1788	Conjunction	4	0%	0%	0	0	0	0%
1791	Nouns - genetives	8	0%	0%	0	0	0	0%
1793	Numeral - inflexions	3	0%	33.3%	0	0	0	0%
1794	Pronouns	14	0%	0%	0	0	0	0%
1796	Adjectives - comparison	13	0%	0%	0	0	0	0%
1797	Confusing "tí" and "ta"	2	0%	0%	0	0	0	0%
1800	Preposition - missing	5	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1801	Preposition	16	0%	6.3%	0	0	0	0%
1807	Subject - missing	8	0%	0%	0	0	0	0%
1808	Word order	2	0%	0%	0	0	0	0%
1810	Important "ið"	47	0%	4.3%	0	0	0	0%
1811	Verb Tense Inconsistent	2	0%	0%	0	0	0	0%
1821	Punctuation - colon	15	0%	0%	0	0	0	0%
1823	Punctuation - semicolon	12	0%	0%	0	0	0	0%
1836	Punctuation - dialog	23	0%	4.3%	0	0	0	0%
1842	Commas - apposition	2	0%	0%	0	0	0	0%
1845	Punctuation - question mark	10	0%	10%	0	0	0	0%
1854	Punctuation - missing space	9	0%	33.3%	0	0	0	0%
1861	Spelling - "vera" and "verða"	48	0%	4.2%	0	0	0	0%
1877	Rep "ks" with "x"	1	0%	0%	0	0	0	0%

ID	Error Type	Num Tests	Correct	Incorrect	$F_{0.25}$	Precision	Recall	Hit
1887	Conjunction - extra	13	0%	0%	0	0	0	0%
1890	Confusing "loyvi" and "loyvt"	1	0%	0%	0	0	0	0%
1891	Confusing "meira" and "fleiri"	4	0%	0%	0	0	0	0%
1892	Confusing "vilja" and "hevði"	3	0%	0%	0	0	0	0%
1894	Rep "mann" with "man"	25	0%	4%	0	0	0	0%
2071	Confusing "lítið" and "fá"	1	0%	0%	0	0	0	0%
2085	Compound words - with or without complement	108	0%	0%	0	0	0	0%
2093	Consonant confusion	5	0%	0%	0	0	0	0%