Goldsmiths
UNIVERSITY OF LONDON

Fine-tuning a pretrained machine learning model to identify declension errors in Icelandic text

Isak Grimsson

Supervisor:

Matthew Yee-King

Goldsmiths

University of London

Department of Computer Science

March 2023

Table of Contents

1 | Abstract

This paper describes the implementation of fine-tuning a pre-trained IceBERT model for the binary classification of declension errors. Our approach includes a method for generating a dataset of incorrectly declined sentences. We provide an overview of the relevant background research, present the results of our model, and discuss their implications, limitations, and potential for future work in this field. Our findings show promising results.

2 | Introduction

> "Icelandic declension is not regular. At all. It does, however, follow patterns, some of which are more common than others. One can find a parallel in English verbs. The majority follow the pattern of love, loved, loved, but there are also irregular groups like swim, swam, swum or wear, wore, wore. It is the vast amount of these patterns in Icelandic that causes problems. As an illustration, Latin, a famously declension-happy language, has twenty declension patterns including irregular nouns. Icelandic has seventy-three."[9]

I began learning Icelandic at the age of seven, today I consider myself a bilingual speaker of English and Icelandic, however there is one element of the Icelandic language that separates me from native Icelandic speakers, declension, as a young child I asked my Icelandic language teacher to explain the logic behind the rules of declension, and discovered there were no explicit rules, only patterns to memorize, ”Read as much as you can, and you will get a feel for it” was the advice I received from my teacher.

Making minor declension errors is common enough even among native Icelandic speakers, but among non-natives, these errors become a constant source of frustration and/or embarrassment. I personally know many non-native Icelandic speakers who face the daily struggle of communicating in a professional environment and having to either accept that their work will have declension errors present or arrange for a native speaker to proofread for them. While there are many spellcheckers available, there is not one that can provide feedback on declension. My hypothesis is that the pattern recognition capabilities of machine learning are a suitable tool to tackle this problem.

In the following chapters I will present to the reader, the areas of relevant background research, the implementation of my solution, the results of my model, a discussion on the implications and limitations of those results, and areas for future work in this space.

3 | Background research

3.1 Subject areas

3.1.1 Machine Learning

AI researcher Arthur Samuel described machine learning as:

> "the field of study that gives computers the ability to learn without being explicitly programmed."[8]

This is an older, informal definition. Well known AI textbook author Tom Mitchell provides a more modern definition:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."[10]

ML can be further broken down into three main categories[12]
- Supervised learning - example inputs with corresponding labelled desired outputs
- Unsupervised learning - no labelled outputs, the model learns patterns without any explicit feedback
- Reinforcement learning - the model learns from reinforcements, rewards and punishments

3.1.2 Natural Language Processing

NLP is a subset of ML which applies machine learning techniques to the analysis and generation of natural language data. To understand what a natural language is, it is useful to first define a formal language, which is a designed language with formal rules, such as programming languages or the language of mathematics or chemistry. A natural

language is a language that has not been designed but rather evolved naturally, and might have some formal rules, but will have many informal or even conflicting rules. Francois Chollet provides an effective definition of Modern NLP in 'Deep learning with Python'

> "Modern NLP is about: using machine learning and large datasets to give computers the ability not to understand language, which is a more lofty goal, but to ingest a piece of language as input and return something useful, like predicting the following:
> - What is the topic of this text? (text classification)
> - Does this text contain abuse? (content filtering)
> - Does this text sound positive or negative? (sentiment analysis)
> - What should be the next word in this incomplete sentence? (language modeling)
> - How would you say this in German? (translation)
> - How would you summarize this article in one paragraph? (summarization)"[2]

3.1.3 Declension

"In Icelandic, nominals (nouns, articles, adjectives, numbers and pronouns) change their form to reflect their function in a sentence." "The process of putting nominals in their various case forms is known as declension"[9] As an example, in the table below are the sixteen variations of the declension pattern that the noun *horse* or *'Hestur'* follows

	Indefinite Article	Definite Article	Indefinite Article	Definite Article
Nominative	Hestur	Hesturinn	Hestar	Hestarnir
Genitive	Hest	Hestinn	Hesta	Hestana
Dative	Hesti	Hestinum	Hestum	Hestunum
Accusative	Hests	Hestsins	Hesta	Hestanna

Table 3.1: Declension of *'Hestur'*

Every Icelandic nominal will fall into one of the 73 declension patterns present in Icelandic, and these patterns are irregular. We hope that readers of this paper who aren't familiar with declension can imagine the difficulty of having to write or speak a sentence where the majority of words change their form based on patterns that evolved from convention rather than from concrete rules.

3.2 Literature review

3.2.1 Transformers

Up until 2017 the ML models used to solve NLP problems would typically either be complex recurrent networks or convolution networks, but the publication of 'Attention is all you need' introduced a new model, which would become the new standard.[2] As the name indicates, the central idea behind the new model is attention.

"An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key."[15]

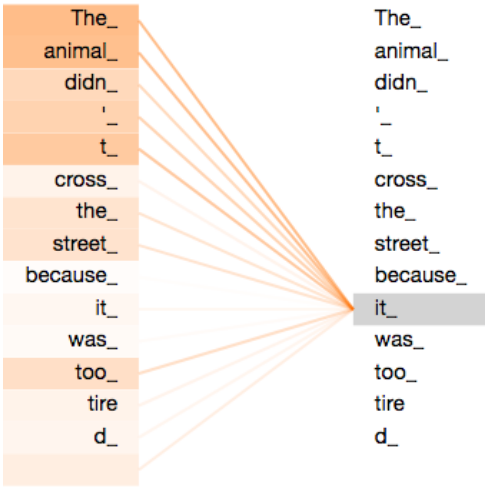

Figure 3.1: Attention visualization

Or as viewed from the layman's perspective, consider the following sentence

"After working too many nights in a row the guard did not enjoy his watch"

the word "watch" can have several meanings: visually observing, a time keeping device, a work shift. How is a model to determine which of these meanings the sentence has? An attention function attempts to solve this by providing the model a mechanism to give a higher weight of importance to context related words, so in our example sentence paying closer attention to the words 'working' 'night' and 'guard' could clue the model into the contextual meaning of the word 'watch'.

Attention functions were not the idea this paper introduced. The highest performing RNN models were already using attention functions. The novel idea presented by the paper was

> "The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution." [15]

The granular explanation of the transformer architecture will be omitted for the sake of brevity, for curious readers I would refer to the original paper or the more digestible four part article series 'Transformers Explained Visually'[6], instead we will focus our

attention on the two major advantages the transformer attention function has over the attention functions used by the previously top performing RNNs.[14]

1. Long-range dependencies

The shift away from processing word by word (sequentially) to processing whole sentences (non-sequentially) solved the challenge of long-range dependencies, as information is not lost as the model loses information (forgets) about previous words.[15]

2. Parallelization via the multi-head attention mechanism

> "Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values"[15]

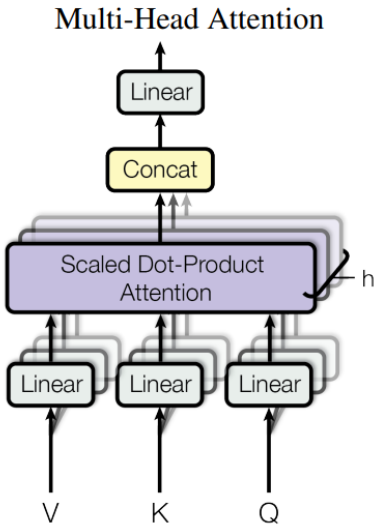

Figure 3.2: MHA visualization

Or in layman's terms: with parallelization the model learns multiple word-to-word relationships and context specific information for each word, and each lesson learnt is then combined, which produces a far richer representation of the data, compared to using 'single-head' attention.

3.2.2 BERT

In 2019 another breakthrough happened in the world of NLP, BERT (Bidirectional Encoder Representations from Transformers) had arrived.

> ”BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks”[4]

It’s built on the transformer architecture and implements masked learning modeling to learn ’bidirectional representation’. In practice this means that 15% of the input words will be unknown to the model, and it will have to guess what word is missing based on the context of the words both to the left and the right of the masked word, self correcting and therefore learning ’bidirectional representation’ over its training cycle.

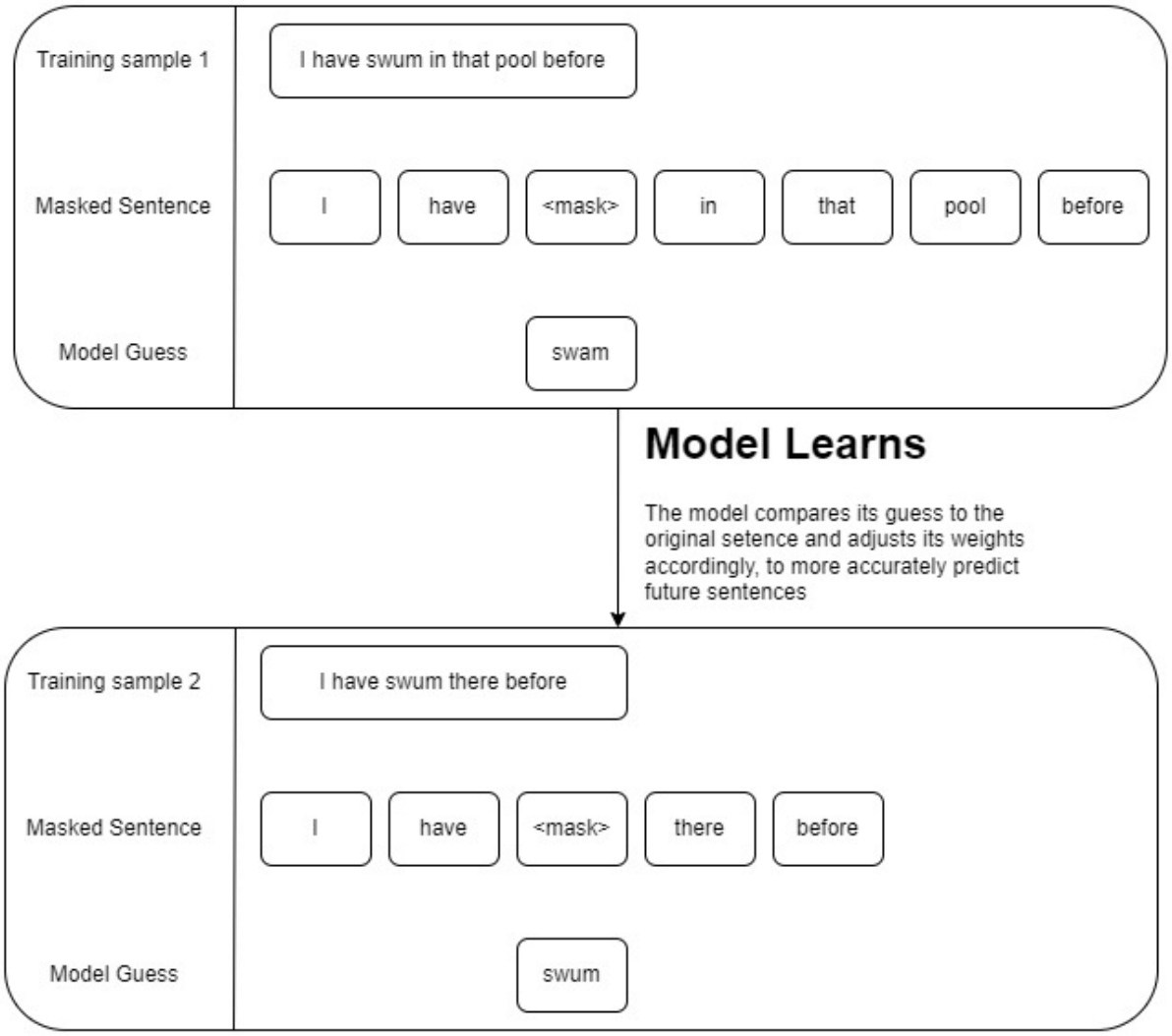

Figure 3.3: The training process of a MLM model

Bert models have been shown to achieve higher performance as the training data set gets larger, they also take longer to train than left to right approaches, as only 15% of the words are predicted in each batch, these two factors make for a slow training process, however that's where the beauty of fine-tuning comes in.

3.2.3 Fine-tuning & pretraining

> "Fine-tuning in NLP refers to the procedure of re-training a pre-trained language model using your own custom data. As a result of the fine-tuning procedure, the weights of the original model are updated to account for the characteristics of the domain data and the task you are interested in."[5]

This means that a very contextually rich model, that took a very long time to train, given the right new data, can be quickly re-trained to perform well on a specific task, combining the benefits of the specific knowledge needed for the task at hand with the deep contextual understanding of the general model. While you could hypothetically fine-tune a BERT model for better performance on a masked language modelling task, in the context of BERT models, fine-tuning is commonly used and referred to as the practice of retraining the model to perform a downstream task such as sentiment analysis or question answering, as opposed to just predicting masked words. In our case we will be attempting to fine-tune it for binary classification, with the purpose of grammar error detection.

3.2.4 IceBERT

IceBERT refers to a Icelandic Bi-Direction Encoder Representation model, trained on a corpus of Icelandic texts from varied sources including, editorial text, a cleaned web crawl, academic papers, published books and historical texts. According to the paper in which it was introduced:

> "achieves state-of-the-art performance in a variety of downstream tasks, including part-of-speech tagging, named entity recognition, grammatical error detection and constituency parsing."[13]

3.3 Identifying a gap of work to be done

The summary of works listed above is that the transformer and later Bert represent giant leaps forward for the world of NLP, due to the ability to quickly retrain models, that have a deep understanding of language, to be used for a variety of tasks. An Icelandic BERT models exists, which means given the right dataset we can fine tune it to perform our specific task, declension error detection. We also have the stretch goal of creating a tool that can perform declension error correction.

3.4 Ethical and social risks

After identifying the gap of work we wish to work on, but before designing and implementing our solution, it's appropriate to consider the ethical and social risks of our project. Google Deepmind published the paper 'Ethical and social risk of harm from Language models' that categorized twenty-one risks spread across six risk areas:

"I. Discrimination, Exclusion and Toxicity
- Social stereotypes and unfair discrimination
- Exclusionary norms
- Toxic language
- Lower performance by social group

II. Information Hazards
- Compromise privacy by leaking private information
- Compromise privacy by correctly inferring private information
- Risks from leaking or correctly inferring sensitive information

III. Misinformation Harms
- Disseminating false or misleading information
- Causing material harm by disseminating misinformation e.g. in medicine or law
- Nudging or advising users to perform unethical or illegal actions

IV. Malicious Uses
- Reducing the cost of disinformation campaigns
- Facilitating fraud and impersonation scams
- Assisting code generation for cyber attacks, weapons, or malicious use
- Illegitimate surveillance and censorship

V. Human-Computer Interaction Harms
- Anthropomorphising systems can lead to overreliance or unsafe use
- Create avenues for exploiting user trust to obtain private information
- Promoting harmful stereotypes by implying gender or ethnic identity

VI. Automation, access, and environmental harms
- Environmental harms from operating LMs
- Increasing inequality and negative effects on job quality
- Undermining creative economies
- Disparate access to benefits due to hardware, software, skill constraints"[16]

For the sake of brevity we will not address each point, but instead address each risk area, but for interested readers I would highly recommend skimming the original paper.

3.4.1 Risk areas I & III

Risk areas I. & III. mostly introduce risks stemming from text generation LMs. There is significantly less risk stemming from this area when working with text-to-text LMs. However the 'lower performance by social group' seems to be a risk that this project has a greater chance of correcting than exacerbating. From a personal perspective, the current grammar correction tools available in Icelandic, seem to me, to be made by Icelandic native speakers, for Icelandic native speakers. The lack of importance placed on functioning declension detection is an oversight that ignores the most common struggle of second language Icelandic speakers, the group that should need grammar correction tools the most. It is my aim to help rectify this oversight.

3.4.2 Risk area II

Risk area II. is not a risk as no data will be collected from the users after development, and all the data to be used for training already exists and is easily accessible.

3.4.3 Risk areas IV & V

Risk areas IV. & V. do not seem to have any obvious connection to a grammatical error detection model.

3.4.4 Risk area VI

Risk area VI. This area introduces the risk of environmental harm, which is unavoidable to some extent due to the nature of how hardware is sourced and manufactured, however if this project were ever to scale up to commercial use, it would only be fitting that it would be hosted in Iceland, powered by environmentally-friendly hydro power energy.[17] I do not foresee this project having a negative effect on creative economies or undermining job quality, however as mentioned above, in risk area I. I do think this project may

increase equality in the job market for second language speakers of Icelandic.

3.4.5 Accessibility

Accessibility is always something to be considered for any software project, if this project is ever made publically available, the user interface will follow guidance from the UK Home Office on designing for accessibility [3].

4 | Implementation

We advise the reader to familiarize themselves with the following abbreviations before continuing to read this chapter.

Abbreviation	The full form
BIN	Database of Icelandic Morphology
GEC	Grammar error correction
GED	Grammar error detection
IEC	Icelandic error corpus

Table 4.1: Frequently used abbreviations

4.1 High level overview of the project

The project consists of three main components:
- A grammar error detection model
- A grammar error correction tool
- and our dataset creation tools

4.1.1 Grammar error detection

We developed a Grammar Error Detection (GED) model by fine-tuning IceBERT for binary sequence classification to distinguish between correctly and incorrectly declined Icelandic sentences.

4.1.2 Grammar error correction

We created a Grammar Error Correction (GEC) tool that uses the GED model to identify declension errors in text and suggests a correction for these errors.

4.1.3 Dataset creation

In order to train and test the GED model we created a tool to parse an existing dataset (IEC), and used that data in conjuction with another dataset (BIN) to create a large synthetic error dataset.

4.2 Requirements and dependencies

4.2.1 IceBERT

IceBERT was defined above in chapter 3.2.4, it was trained on the following data

Dataset	Size	Tokens
IGC(editorial text)	8.2GB	1,388M
IC3(cleaned-webcrawl)	4.9GB	824M
Student theses	2.2GB	367M
Greynir News articles	456MB	76M
Medical library	33MB	5.2M
Open Icelandic e-books	14MB	2.6M
Icelandic Sagas	9MB	1.7M
Total	15.8GB	2,664M

Table 4.2: IceBERT training data

In the paper in which it was introduced, its results once fined tuned for GED were

Model	F1	Precision	Recall	Accuracy
IceBERT	70.11%	92.30%	56.53%	96.99%

Table 4.3: IceBERT results after fine-tuning for GED

4.2.2 Data

4.2.2.1 IEC

A collection of over 3600 XML files, containing text that had been proofread and corrected, along with annotations on the type of correction made.The text consists of proofread essays, Wikipedia entries and news articles.

4.2.2.2 BIN

A database which contains over 300,000 unique words morphing into over 6,500,000 different cases, depending on declension, definite article and plurality.

4.2.3 Tech stack

4.2.3.1 Python

We chose to write our code in Python due to its powerful string manipulation methods as well as its ubiquitous presence in ML projects, resulting in a large community of developers providing support, resources, libraries and frameworks aimed towards ML projects.

4.2.3.2 Jupyter notebook

All of our code was written on Jupyter notebook, due to its ease of collaboration and ability to reproduce results, it also has a ubiquitous presence in ML projects.

4.2.3.3 Hugging Face

We used the Hugging Face library, which is an NLP library built on top of PyTorch and TensorFlow[7], it was used to pretrain IceBERT, so it was practical for us to continue using it for fine-tuning.

4.2.3.4 Pandas

Pandas is a python library, often used for ML projects due to its high-performance analysis and data manipulation tools for structured data[11]. It played an essential role in creating and manipulating our datasets.

4.2.3.5 Beautiful Soup

Beautiful Soup is a Python library for parsing HTML and XML documents. It provides a simple way to extract data from web pages and offers features for navigating, searching, and modifying the parsed document tree[1]. We utilized it to preprocess the IEC.

4.3 IEC_Parser

The first dataset we trained our model with was the IEC, as defined above in chapter 4.2.2.1. The IEC is a collection of proofread XML files. To preprocess the data into an appropriate format, we built the IEC_Parser which extracts each sentence from the XML files and adds them to a CSV file, along with a label of correct/incorrect, and if incorrect, the type of error corrected.

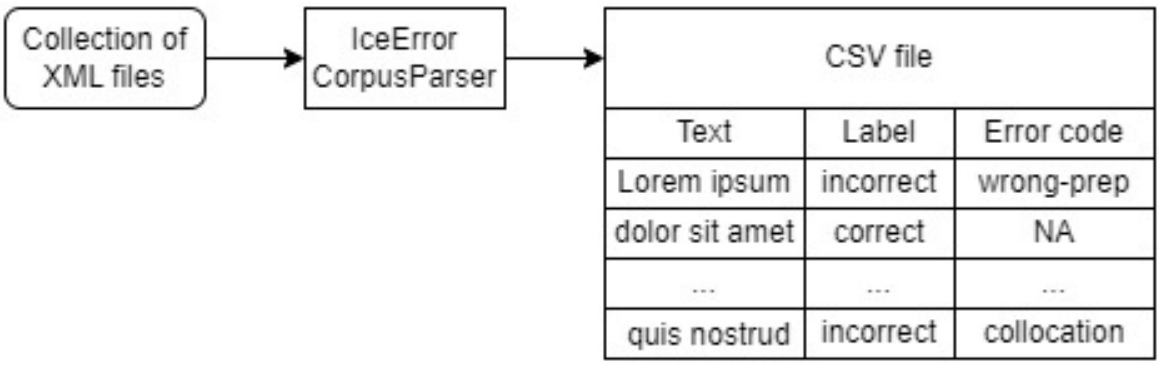

Figure 4.1: Input to output of IEC_Parser

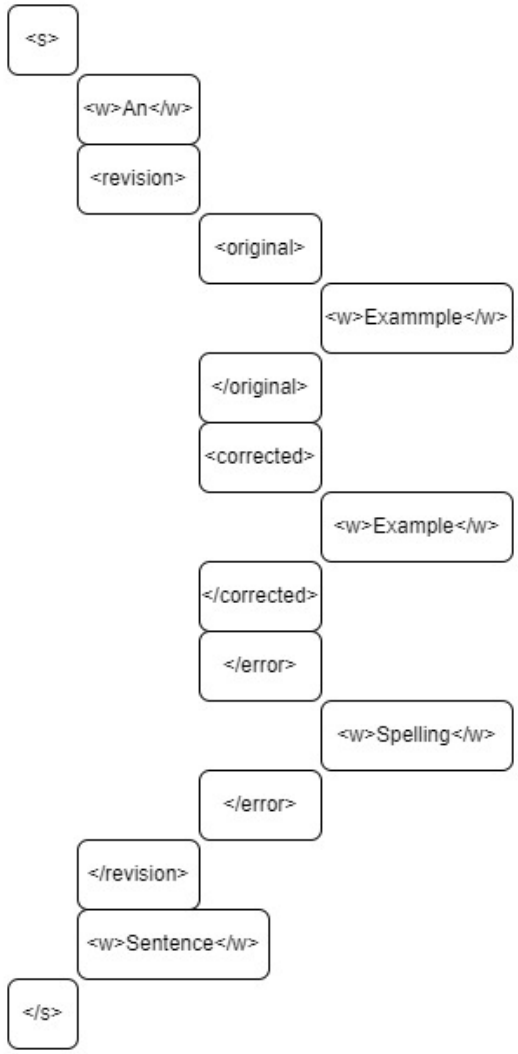

Figure 4.2: An example XML sentence

Each sentence of the XML file is structured like the example shown in figure 4.2. The BeautifulSoup library allowed us to efficiently select the target content of each sentence, i.e.

"An exammple spelling sentence",
incorrect, spelling

We trained the model on this dataset, which produced quite lackluster results.[a] The IEC has a sample size of around 35,000 correct sentences, and 22,910 incorrect samples, but that number drops down to just over 1000 if only declension errors are counted. We hypothesized that our model needed more data, and so we looked to combine the IEC dataset with the BIN dataset to create more data.

[a] the results for this model will be discussed in chapter 5

4.3.1 BIN_Parser

To address the lack of data, we created synthetic data using BIN, a database which contains over 300,000 unique words morphing into over 6,500,000 different cases, depending on declension, definite article and plurality. In Icelandic, a noun in any given sentence can only have one correct form, therefore since we have a dataset of 35,000 correctly declined sentences, and another database with all the various declension cases of the nouns

found in those sentences, it is possible to create a synthetic dataset containing incorrect declensions. Consider the Icelandic sentences "Hann er hestur" (He is a horse) and "Hann er kind" (He is a sheep) these are both grammatically correct sentences, however if you substitute the noun of either sentence out with a different declension case, i.e. "Hann er hest" or "Hann er kindar" they would be grammatically incorrect. Using this knowledge, if given a correct sentence, we can create multiple incorrect sentences for each different noun in that sentence. Using this method in conjunction with the 35,000 correct sentences from the IEC, we created a synthetic dataset of over 850,000 incorrectly declined sentences.

	Horse	Sheep
Nominative	Hestur	Kind
Genitive	Hest	Kind
Dative	Hesti	Kind
Accusative	Hests	Kindar

Table 4.4: Declension of *'Hestur'* and *'Kind'*

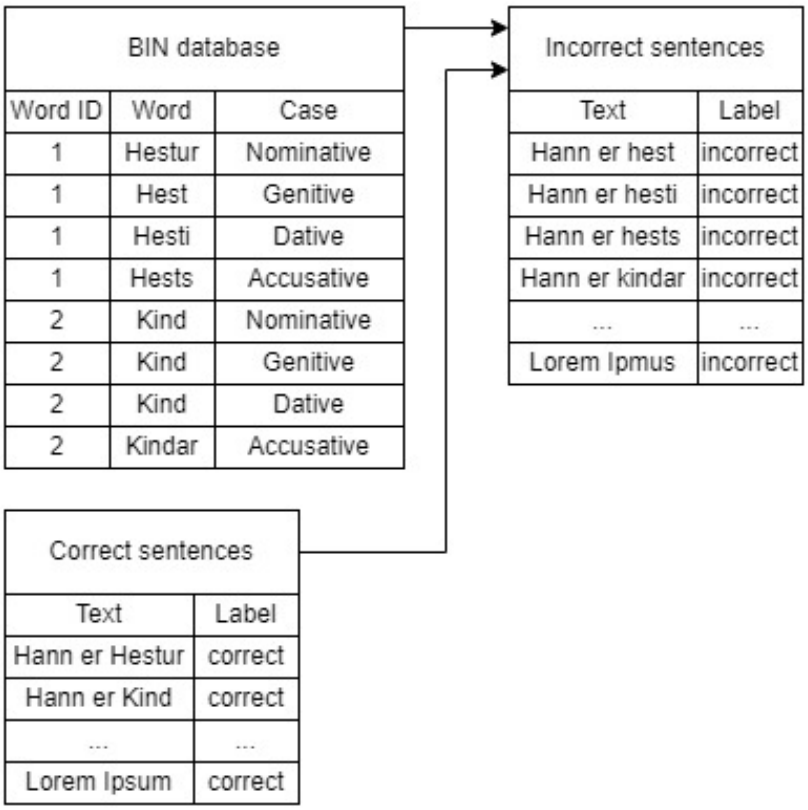

Figure 4.3: Sentence generation with incorrectly declined nouns, using BIN_Parser

However this only describes the generation of sentences with incorrect nouns, and if Icelandic declension only affected nouns, we could stop here, but we also need to consider the declension of all the following nominals: verbs, adjectives, pronouns and numerals. Figure 4.4 describes the steps we follow to generate incorrect sentences for these nominals, however for the sake of brevity, we will omit an explanation of how we ensure incorrect sentences for each of the categories, but all follow a similar pattern as the one explained for nouns. Using this method we created a dataset of over 1,000,000 incorrectly declined sentences.

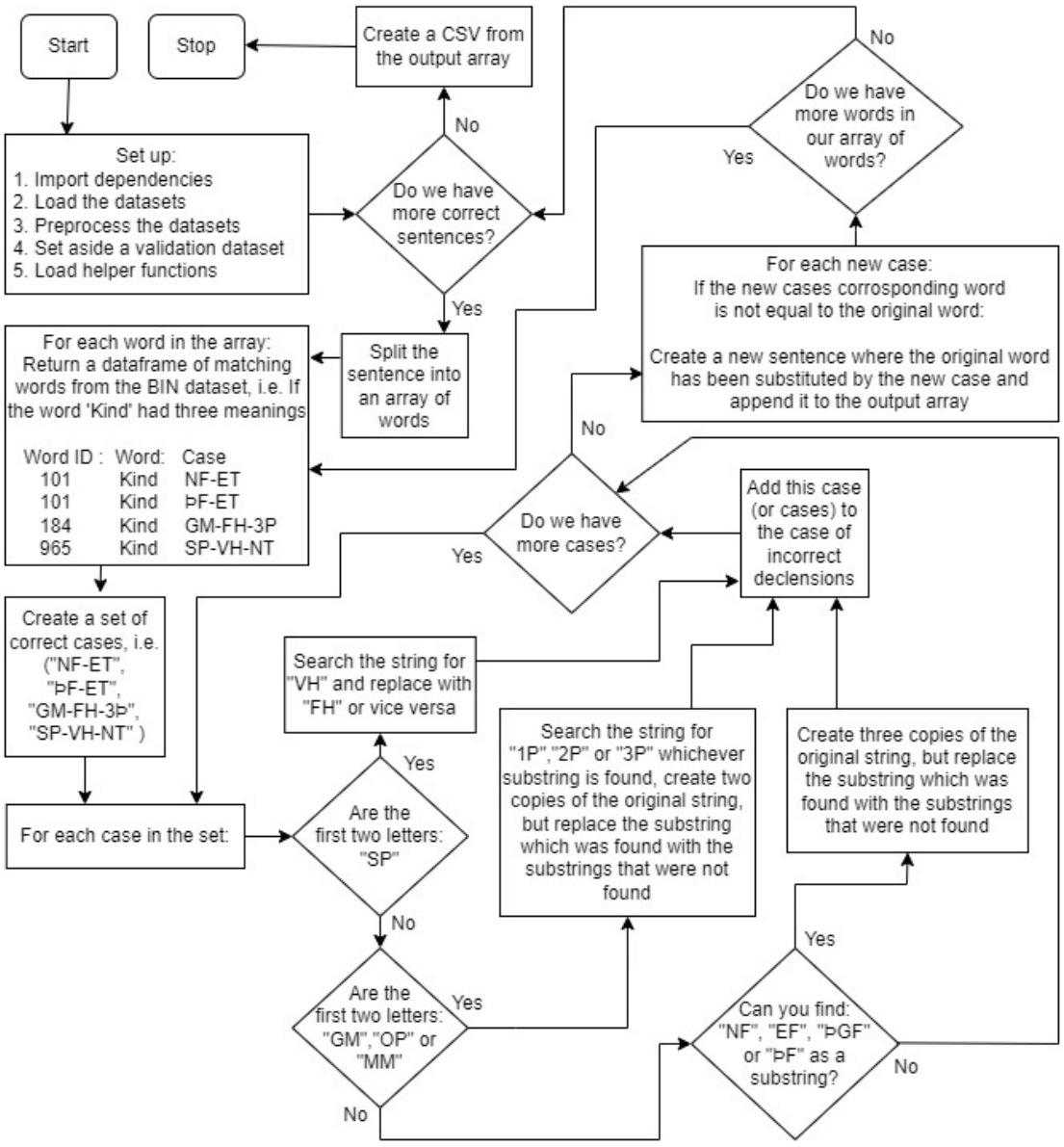

Figure 4.4: Flowchart of BIN_parser

4.4 Model implementation

4.4.1 Grammatical error detection

Using the AutoModels function of the HuggingFace NLP library, we are able to import the pretrained IceBERT model and apply the relevant tokenizer, model and classifier in just three lines of code.

```
# Model Architecture
myTokenizer = AutoTokenizer.from_pretrained("mideind/IceBERT")
myModel = AutoModelForSequenceClassification.from_pretrained(
    "mideind/IceBERT", num_labels=2, id2label=id2label, label2id=label2id)
classifier = pipeline("text-classification", model=myModel, tokenizer=myTokenizer)
```

Figure 4.5: Model Architecture

Then we need to prepare our model for training by loading and preprocessing our dataset into the format the model is expecting

```
# Load our two datasets
# 1. IceErrorCorpus (IEC)
correct_data = pd.read_csv("./generated_datasets/labeled_data.csv", encoding='latin-1')
correct_data = correct_data.loc[correct_data['label'] == 'correct']
correct_data = correct_data.drop(['Error'], axis=1)
```

```
# 2. Our synthetic dataset generated from IEC and BIN
error_data = pd.read_csv("./generated_datasets/synthetic_data.csv", encoding='UTF-8')
error_data = error_data.drop(['type'], axis=1)
error_data = error_data.drop(['error position'], axis=1)
error_data  = error_data.drop_duplicates()
```

```
# Combine the correct and synthetic datasets
data = pd.concat([correct_data, error_data])
# Replace  the labels with numeric values
data['label'] = data['label'].replace(['correct','incorrect'],[0,1])
# Split the data into training and testing sets
train, test = train_test_split(data, test_size=0.2)
```

Figure 4.6: Loading and preprocessing our datasets

after which we set a few training arguments such as learning rate, batch size and

number of epochs, before simply running the trainer function to retrain the model. After the model has retrained we can use the classification function to pass in a string of text, and receive a binary classification from the model of either 'correct' or 'incorrect' as well as a confidence score of its classification.

4.4.2 Grammatical error correction

For our GEC tool we start by loading our retrained model and our BIN dataset, and reading in a CSV file of sentences to be corrected. We then feed each sentence of the CSV file into the model classifier one at a time, if we detect an error, for each word of the sentence, we search the BIN database for alternate cases of said word, and for each alternate case found, we create a new sentence, where the original word has been replaced with the alternate case, and we send that sentence through the classifier.
While classifying these alternate cases, we collect any sentences classified as correct in an array, and after classifying all the possible case variations of the original sentence, we return the sentence with the highest confidence score as our suggested correction.

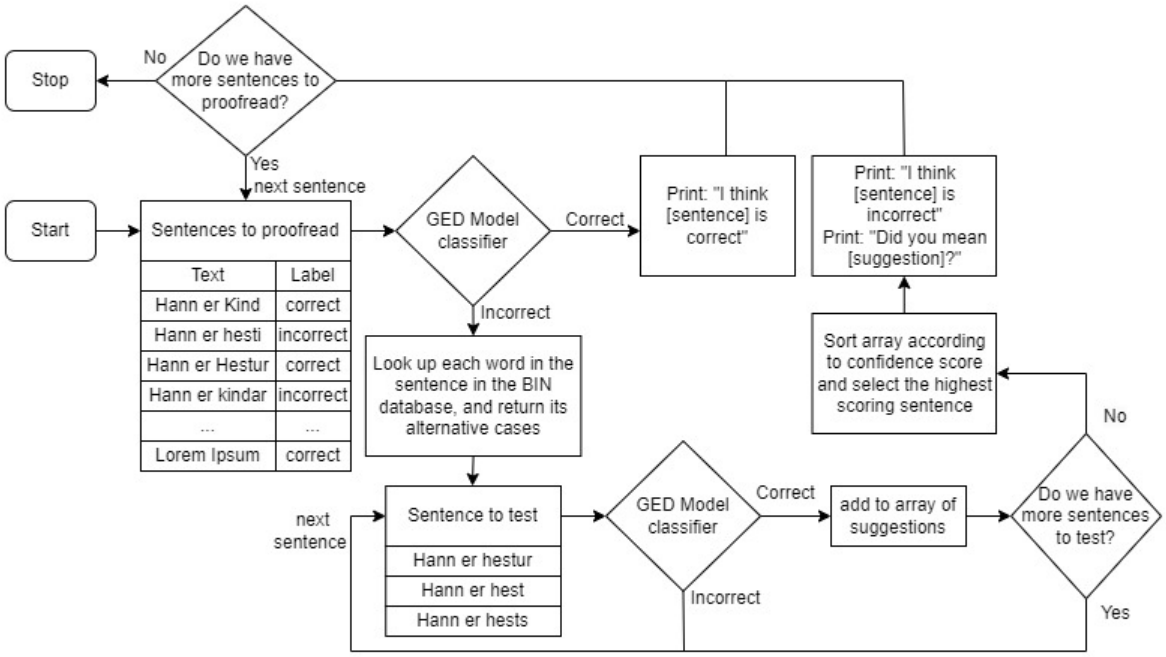

Figure 4.7: GEC flowchart

5 | Results

The GED model and the GEC tool require separate evaluations, we will start by evaluating the GED model, i.e. how well does it perform at classifying a sentence as correct or incorrect?

5.1 Training Validation

Up until this point, this paper has been focused on the final implementation of the project, however many different datasets and models were tested during the iterative process towards the project's current state. Below table 3.1 presents the major milestones of development.

Model Name	Description	Correct	Incorrect
IEC Unfiltered	IEC Dataset	35,000	25,000
IEC Filtered	IEC Dataset filtered to only include declension errors	35,000	1,000
Small synthetic nouns	Correct sentences from IEC and sentences of incorrectly declined nouns generated from IEC and BIN	5,000	5,000
Large synthetic nouns	Same as above	35,000	468,000
Small synthetic nominals	Correct sentences from IEC and sentences of incorrectly declined nominals generated from IEC and BIN	35,000	57,000
Mid synthetic nominals	Same as above	35,000	135,000
Large synthetic nominals	Same as above	35,000	443,000
Giant synthetic nominals	Same as above	35,000	1,659,000

Table 5.1: Major milestones of model development

Below in table 5.2, are the metrics of these different models up to this point. The meaning of the columns is as follows:

- Epochs, the number of epochs run to achieve minimum validation loss

- Training & Validation, training/validation loss at the specified epoch

- Correct & Incorrect, as a sanity check we classified 10 sentences the model had never seen before, five that were incorrect and five that were correct, these metrics reflect how many the model correctly classified.

Model Name	Epochs	Training	Validation	Correct	Incorrect
IEC Unfiltered	2	0.6239	0.5483	5/5	2/5
IEC Filtered	1	0.2438	0.2514	5/5	0/5
Small synthetic nouns	2	0.6271	0.2794	5/5	5/5
Large synthetic nouns	1	0.08220	0.08044	5/5	5/5
Small synthetic nominals	2	0.1333	0.1784	5/5	5/5
Mid synthetic nominals	1	0.1695	0.1496	4/5	5/5
Large synthetic nominals	2	0.2044	0.1922	0/5	5/5
Giant synthetic nominals	1	0.1017	0.1018	0/5	5/5

Table 5.2: Training metrics of our models

The first two models in the tables above, were experimental models, quickly trained and quickly discarded. The small synthetic nouns model goal was to test the feasibility of training on synthetic data, when it correctly classified all ten sanity check sentences it marked an exciting milestone. The subsequent large synthetic nouns model was the first model to perform well on the training validation data, providing tangible evidence for the model's effectiveness, at which point the next steps to take were clear: expand the model to consider not just nouns but all nominals, as well as test the models performance on an unseen validation set.

5.2 Validation dataset

In order to get a clear idea of how well a model performs, we need to test it on data it has never seen before. In order to do this we created a validation set consisting of 4495 sentences from the IEC, 1036 sentences that had been labelled as incorrect declension, and 3549 correct sentences. All of these sentences were removed from the dataset before it was used to create the synthetic data and train the model.

5.3 Validation metrics

To validate our model we will be measuring its accuracy, precision, recall and f1 score. A correct sentence will be labelled as "positive", and an incorrect sentence will be labelled as "negative".

		Actual class	
		Positive	Negative
Predicted Class	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 5.1: A confusion matrix

$$accuracy : \left(\frac{TN + TP}{TN + FP + TP + FN}\right)$$

$$precision : \left(\frac{TP}{TP + FP}\right)$$

$$recall : \left(\frac{TP}{TP + FN}\right)$$

$$f1 - score : \left(2 * \frac{precision * recall}{precision + recall}\right)$$

The f1 score is the most important metric we will use to judge our models accuracy, as it takes into account the model's performance on both the majority and minority classes, which is very important when evaluating imbalanced datasets.

5.4 Validation results

5.4.1 ChatGPT-4

At the start of this project, there were no available tools capable of detecting declension errors, but during development, ChatGPT-4 was released, which added Icelandic to the list of languages it can understand. As a benchmark we would like to compare our results with the results of ChatGPT-4 when asked to detect declension errors from our validation set. The prompt used to gather these classifications can be found in the appendix.

		Actual class	
		Positive	Negative
Predicted Class	Positive	2691	693
	Negative	768	343

Figure 5.2: ChatGPT-4's confusion matrix

$$accuracy : 0.6750$$

$$precision : 0.7952$$

$$recall : 0.7779$$

$$f1 - score : 0.7865$$

5.4.2 Large synthetic noun model

Consisted of approximately 468,000 incorrect and 35,000 correct sentences.

		Actual class	
		Positive	Negative
Predicted Class	Positive	3169	310
	Negative	290	726

Figure 5.3: The large noun model's confusion matrix

$accuracy$: 0.8665

$precision$: 0.9109

$recall$: 0.9161

$f1 - score$: 0.9135

5.4.3 Small synthetic nominal model

Consisted of approximately 57,000 incorrect and 35,000 correct sentences.

		Actual class	
		Positive	Negative
Predicted Class	Positive	3430	556
	Negative	29	480

Figure 5.4: The small nominal model's confusion matrix

$accuracy$: 0.8699

$precision$: 0.8605

$recall$: 0.9916

$f1 - score$: 0.9214

5.4.4 Mid synthetic nominal model

Consisted of approximately 135,000 incorrect and 35,000 correct sentences.

		Actual class	
		Positive	Negative
Predicted Class	Positive	3363	482
	Negative	96	554

Figure 5.5: The mid nominal model's confusion matrix

$accuracy : 0.8714$

$precision : 0.8746$

$recall : 0.9722$

$f1 - score : 0.9209$

5.4.5 Large synthetic nominal model

Consisted of approximately 468,000 incorrect and 35,000 correct sentences.

		Actual class	
		Positive	Negative
Predicted Class	Positive	0	0
	Negative	3459	1036

Figure 5.6: The large nominal model's confusion matrix

$accuracy : NA$

$precision : NA$

$recall : NA$

$f1 - score : NA$

5.4.6 Giant synthetic nominal model

Consisted of approximately 1,640,000 incorrect and 35,000 correct sentences.

		Actual class	
		Positive	Negative
Predicted Class	Positive	0	0
	Negative	3459	1036

Figure 5.7: The giant nominal model's confusion matrix

$$accuracy : NA$$

$$precision : NA$$

$$recall : NA$$

$$f1 - score : NA$$

5.4.7 Validation result comparison

Note: our last two models scores cannot be calculated as the number of true positives are 0. We cannot divide by zero to obtain the score, per the formula in chapter 5.3.

Model Name	Accuracy	Precision	Recall	F1-score
Large synthetic nouns	0.8665	0.9109	0.9161	0.9135
Small synthetic nominals	0.8699	0.8605	0.9916	0.9214
Mid synthetic nominals	0.8714	0.8746	0.9722	0.9209
Large synthetic nominals	NA	NA	NA	NA
Giant synthetic nominals	NA	NA	NA	NA

Table 5.3: The validation scores of our models

5.5 GEC Evaluation

To evaluate the GEC tool we use the 1000 incorrectly declined sentences, with labelled corrections, found in the IEC dataset. We feed the GEC tool the incorrect sentence, and compare the suggestion it returns to the corrected sentence provided by the IEC.

Model Name	Match	Mismatch	No suggestion	Misclassified
Large synthetic nouns	11.46%	27.55%	31.02%	29.96%
Small synthetic nominals	12.14%	24.88%	9.63%	53.76%
Mid synthetic nominals	11.75%	25.91%	15.70%	46.63%
Large synthetic nominals	0%	0%	100%	0%
Giant synthetic nominals	0%	0%	100%	0%

Table 5.4: Training metrics of our models

6 | Discussion

6.1 Analysis of the GED model

Our best scoring model has an F1-score of 0.9214, which is a significant performance improvement compared to ChatGPT-4's F1 score of 0.7865. These are quite exceptional results, declension error detection can be a tricky task even for native speakers of Icelandic, the validation set was made up of sentences from the IEC dataset, which as mentioned in chapter 4.2.2.1 consists of essays, wikipedia entries and news articles written by presumably mostly native Icelandic speakers. It is the author's opinion that based on how it performed on that dataset it seems all but certain that the model already outperforms the vast majority of non-native Icelandic speakers, and it might even rival many native speakers. It would be interesting to test these claims as mentioned in the future works chapter 7.1.8.

Our first three models all have a similar f1-score, with the large noun model having the best precision, while the synthetic models had lower precision but a far better recall score. The last two models classified every sentence as incorrect. It can be deduced that they simply learnt that their training set was imbalanced enough that if they classified every sentence as incorrect, they would be right almost every time. However it's interesting to note that the large noun model had a marginally more imbalanced dataset than the large nominal model, and it didn't learn the same lesson, it seemed to perform well on unseen data, as seen by both the sanity check results, and its validation test results. These results encourage further experimentation, perhaps the model is capable of learning the rules of noun declension, but it's unable to distinguish a rule set to the different declension rules for all nominals at the same time. We will discuss what could be done to test this in chapter 7.1.4.

6.2 Analysis of the GEC model

The misclassification error rate of the models is a measure of the GED metric so we will forgo commenting on that in this section, however the other three metrics paint a picture of the limitations of the GEC tool. The first three models displayed in table 5.3 all tell a similar story in regards to matches and mismatches, an exact match being between 11-12% of cases and a mismatch being between 24-27%. The low match rate is due to how the tool functions, it replaces one word at a time with possible alternative cases, so in cases where there were multiple corrections to be made, or the order of the original sentence was changed, or any other change to the sentence structure that wasn't limited to a one word change, the tool will always fail to find an exact match, resulting either in a "No suggestion" where it cannot find any sentence it deems to be grammatically correct, or a mismatched sentence where the tool finds a sentence it deems to be suitable, but isn't the exact same sentence provided in the proofread dataset. This has lead us to two conclusions, firstly that judging the success of a GEC model by if it returns an exact match is a flawed approach as there are often many different ways to correct a sentence, and secondly that the single word substitute method is too naive of an approach for an effective GEC model, and that an entirely new design is needed. To be an effective tool it cannot simply try to substitute words, we suspect it would need to be able to understand the meaning of the input text, and generate a new grammatically correct sentence from that meaning, we mention our desire to explore this further in chapter 7.2.

7 | Limitations and future works

7.1 GED Model

7.1.1 Cross validation

Cross validation would enable our model to train with our entire dataset and ensure that the validation score doesn't get a lucky or unlucky score. However this would have multiplied the already significant training time of our models, so we opted to simply split our data into 80% training and 20% test.

7.1.2 Model parameter fine-tuning

Ideally we would have experimented with fine-tuning each model's parameters, but similar to the point above, this would have significantly increased the training times, so we opted to stick to the parameters used by the pre-trained IceBERT model.

7.1.3 ROC & AUC implementation

Calculating the 'Area Under the Curve' of the 'Receiver Operating Characteristic' would have allowed us to identify the ideal threshold to classify a sentence as correct or incorrect, but time did not allow for this.

7.1.4 Divide & conquer

As mentioned in chapter 4.1, based on the validation tests we would like to experiment by dividing up the training data into separate datasets for each nominal category, and train models for each nominal class and see how they function individually, and possibly as a chain of classifiers.

7.1.5 Source or generate correct sentences

Using BIN parser we can generate dozens of incorrect example sentences for every correct sentence, so the data bottleneck is how many correct sentences we can source or generate. It would be an interesting project to see if there is a way to create a dataset of reliably declined Icelandic sentences.

7.1.6 Creating a user interface

Creating a user interface, most likely a website, so the model can be used.

7.1.7 Human performance comparison

It would be interesting to compare the models performance to a group of both native and non-native Icelandic speakers.

7.2 GEC Tool

A future project could be to design a more robust GEC solution, perhaps better results could be achieved by combining the detection capabilities of the GED model with a text generation model such as Chat-GPT4.

8 | Conclusion

In this paper we have presented the implementation and results of our GED model and GEC tool. In regards to the GEC tool, while it's our observation that the tool does function better in general practice than its strict one-to-one comparison metrics indicate, its limitation to only correcting one word at a time has been shown to not be a robust solution, and it is our opinion that a different approach would be needed to satisfactorily solve this problem.

As for the GED model, the results have been extremely promising, but even so there is still room for much improvement as was outlined in chapter 7.1. To call back to our original hypothesis from chapter 1. we believe this paper has proven that ML is a suitable tool to tackle the problem of irregular declension patterns, and we are proud of the contribution it will make to the Icelandic NLP domain.

Bibliography

[1] *BeautifulSoup*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visited on 03/11/2023).

[2] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[3] *Designing for Accessability*. URL: https://github.com/UKHomeOffice/posters/blob/master/accessibility/dos-donts/posters_en-UK/accessibility-posters-set.pdf (visited on 12/12/2010).

[4] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] *fine tuning for domain adaptation in nlp*. URL: https://towardsdatascience.com/fine-tuning-for-domain-adaptation-in-nlp-c47def356fd6) (visited on 09/30/2010).

[6] *How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models*. URL: https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452 (visited on 09/30/2010).

[7] *HuggingFace*. URL: https://huggingface.co/about (visited on 03/11/2023).

[8] *Icelandic noun cases and declensions*. URL: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained#:~:text=Machine%20learning%20is%20one%20way,learn%20without%20explicitly%20being%20programmed.%E2%80%9D (visited on 05/03/2023).

[9] Daisy L Neijmann. *Colloquial Icelandic: the complete course for beginners*. Routledge, 2006.

[10] Andrew Ng. *Standford Machine Learning*. URL: https://www.coursera.org/learn/machine-learning (visited on 09/30/2010).

[11] *Pandas*. URL: https://pandas.pydata.org/about/ (visited on 03/11/2023).

[12] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[13] Vésteinn Snæbjarnarson et al. "A Warm Start and a Clean Crawled Corpus–A Recipe for Good Language Models". In: *arXiv preprint arXiv:2201.05601* (2022).

[14] *understanding transformers nlp state of the art models*. URL: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/ (visited on 09/30/2010).

[15] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[16] Laura Weidinger et al. "Ethical and social risks of harm from language models". In: *arXiv preprint arXiv:2112.04359* (2021).

[17] *Why Icelandic data centres are the 'greenest in the world'*. URL: https://www.itpro.co.uk/server-storage/data-centres/369253/why-icelandic-data-centres-are-the-greenest-in-the-world (visited on 12/16/2010).

A | Web Links

A.1 Code

Github: `https://github.com/Isak14/declension_error_detection_model`

A.2 Datasets

Ice Error Corpus: `https://github.com/antonkarl/iceErrorCorpus`

BIN: `https://bin.arnastofnun.is/DMII/LTData/data/mimisbrunnur/`

A.3 Models

Small synthetic nominals: `https://huggingface.co/IsakG/declension_error_detection`

B | User guide

The github repository linked above has six jupyter notebook files, each of them include markdown comments to explain each functional step, as well as code comments to explain the code in detail. The model and datasets were too large to upload to Github, however our best performing model (small synthetic nouns) was uploaded to HuggingFace.
To use it, simply follow the link above, enter in the sentence you wish to classify on the right, and it will return its classification.

C | List of Abbreviations

AI: Artificial Intelligence

BERT: Bidirectional Encoder Representations from Transformer

BERT: Bidirectional Encoder Representations from Transformer

BIN: Beygingarlysing islensks nutimamals (Database of Icelandic Morphology)

CNN: Convolutional neural network

CSV: Comma-separated values

GEC: Grammar error correction

GED: Grammar error detection

HTML: HyperText Markup Language

IEC: Icelandic error corpus

IceBERT: Icelandic Bidirectional Encoder Representations from Transformer

LMs: Language Models

MHA: Multi Head Attention

ML: Machine learning

MVP: Minimum Viable Product

NLP: Natural language processing

RNN: Recurrent neural network

RoBERTa:A Robustly Optimized Bidirectional Encoder Representations from Transformer pretraining approach

UX: User Experience

XML: Extensible Markup Language