MASTER'S THESIS 2024

Building a Framework for Evaluation of Small Language Models: A Study of NLP Task Performances and Swedish Comprehension

Amanda Axelsson, Allis Rodvaldr

ISSN 1650-2884

LU-CS-EX: 2024-37

DEPARTMENT OF COMPUTER SCIENCE

LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-37

Building a Framework for Evaluation of Small Language Models: A Study of NLP Task Performances and Swedish Comprehension

Skapandet av ett ramverk för evaluering av små språkmodeller: En studie i naturlig språkbehandling och svensk språkförståelse

Amanda Axelsson, Allis Rodvaldr

Building a Framework for Evaluation of Small Language Models: A Study of NLP Task Performances and Swedish Comprehension

Amanda Axelsson
manda.axelsson@hotmail.com

Allis Rodvaldr
allis@rodvaldr.se

June 25, 2024

A thesis presented for the degree of
Master of Science in Engineering Physics

Department of Computer Science, Lund University.

Supervisors: Maj Stenmark, maj.stenmark@cs.lth.se
Henrik Tingström, henrik@scholaro.io

Examiner: Elin A. Topp, elina.topp@cs.lth.se

Abstract

As of today, computational resources are one of the biggest limitations to the development and deployment of natural language processing (NLP) models in organizations. For Scholaro, a company that aims to build a scholarship matching platform driven by NLP models, the extensive usage of large language models (LLMs) is becoming difficult to financially motivate. Therefore, this thesis explores whether small language models (SLMs) could offer a more cost-effective alternative to LLMs, and in that case, to what extent. The effectiveness of five SLMs for both bilingual (English and Swedish) as well as task-specific applications, including classification, summarization, and translation is being studied. Furthermore, the study employs a framework that introduces a variety of human and automatic evaluation metrics suitable for measuring the performance of these SLMs. The findings suggest that SLMs can, if the correct model, prompt, and task are combined, serve as viable alternatives to LLMs, particularly in scenarios where resource efficiency and task specificity are vital.

Keywords: natural language processing, machine learning, artificial intelligence, scholarships, small language models, large language models, multi-label classification, summarization, translation, Swedish, evaluation

Acknowledgements

We want to thank our supervisor at Scholaro, Henrik Tingström, for continuous support during the process. His enthusiasm regarding the project has been proven invaluable.

We also want to thank our supervisor at LTH, Maj Stenmark, who has kept us on the right track throughout the whole process, been supporting, and able to answer all kinds of questions.

Finally, thanks to our examiner Elin A. Topp, for being so accommodating and helpful towards the end of our project.

Contents

1 Introduction **9**

1.1 Context . . . 9
1.2 Scholaro . . . 10
1.3 Problem formulation and research questions . . . 10
1.4 Related work . . . 11
1.4.1 The performance of small language models . . . 11
1.4.2 Evaluation of small language models . . . 12
1.4.3 Swedish in language models . . . 13
1.5 Contributions . . . 13
1.5.1 Contribution to research . . . 13
1.5.2 Division of work between authors . . . 14
1.6 Limitations . . . 14
1.7 Outline of report . . . 15

2 Background **17**

2.1 Machine learning . . . 17
2.2 Artificial neural networks . . . 17
2.2.1 Loss function . . . 18
2.2.2 Network architecture . . . 18
2.2.3 Activation function . . . 20
2.2.4 Feed-forward and recurrent networks . . . 20
2.3 Transformer models . . . 20
2.3.1 Tokenization . . . 21
2.3.2 Word embedding . . . 22
2.3.3 Positional encoding . . . 24
2.3.4 The transformer block . . . 24
2.3.5 Output layer . . . 25

2.4 Large language models . . . 25
2.4.1 Applications and limitations . . . 26
2.4.2 Implementation costs . . . 26
2.4.3 Environmental impact . . . 26
2.5 Small language models . . . 28
2.5.1 Why SLM instead of LLM . . . 28
2.6 Prompt engineering . . . 28
2.6.1 COSTAR prompting framework . . . 29
2.6.2 Prompting SLMs using LLMs . . . 29
2.6.3 Few-shot learning . . . 29
2.6.4 Chain-of-thought prompting . . . 29
2.7 Model evaluation . . . 30
2.7.1 Automatic evaluation . . . 31
2.7.2 Human evaluation . . . 35

3 Method **37**
3.1 The dataset . . . 37
3.1.1 Data gathering . . . 37
3.1.2 Preprocessing of dataset . . . 38
3.1.3 Labeling of dataset . . . 39
3.2 The models . . . 40
3.2.1 Mistral . . . 41
3.2.2 StableBeluga . . . 41
3.2.3 OpenOcra . . . 42
3.2.4 Qwen . . . 42
3.2.5 GPT-SW3 . . . 42
3.3 Prompt engineering . . . 43
3.3.1 COSTAR prompting framework . . . 43
3.3.2 Prompting SLMs using LLMs . . . 44
3.3.3 Prompting tags . . . 44
3.3.4 Few-shot learning . . . 44
3.3.5 Chain-of-thought prompting . . . 44
3.4 Coding details . . . 46
3.4.1 Python libraries . . . 46
3.4.2 Langchain framework . . . 46
3.5 Model evaluation . . . 47
3.5.1 Automatic evaluation . . . 48
3.5.2 Human evaluation . . . 49
3.5.3 Cost-effectiveness . . . 50

4 Result **53**
4.1 Table 53
4.2 User survey comments 55

5 Discussion **57**
5.1 Prompting and usability 57
5.2 Evaluation of the models 58
5.2.1 Benchmarking against GPT-4 58
5.2.2 Classification 58
5.2.3 Summarization 61
5.2.4 Translation 63
5.2.5 Perplexity 64
5.2.6 Cost-effectiveness and resource use 65
5.3 Future research 67
5.3.1 Energy consumption 67
5.3.2 Data privacy 68
5.3.3 Data governance 68
5.3.4 Bias 68
5.3.5 Fine tuning with custom dataset 69
5.3.6 Combination of several SLMs 69

6 Conclusion **71**

Appendix A The labeled dataset **83**

Appendix B The scripts in Python **85**

Appendix C Prompts for all three tasks **93**

Appendix D User survey participants **97**

Chapter 1

Introduction

This chapter introduces the context of the thesis, including details about the collaborating company, Scholaro. It also presents the research questions and an assessment of current related work in the field. Lastly, it discusses the contributions to research, the division of work between the authors, and the project's limitations.

1.1 Context

A challenge facing Sweden today concerns the financial well-being of students. For example, in Stockholm, the expenses for an average student exceed their income by 4,585 SEK each month, according to Stockholms studentkårer (SSCO). With finances under strain, students are compelled to seek additional income during their studies to make ends meet. Consequently, it becomes apparent that many students would greatly benefit from an extra income [1].

One potential way to address this issue is through scholarships, a tax-exempt financial support provided by various organizations, foundations, and companies. Scholarships can range from a few thousand SEK to amounts sufficient to cover the costs of an entire educational program abroad. However, securing a scholarship requires locating it and submitting an application, a process that can be long, difficult, and sometimes fruitless. Additionally, some students are simply unaware of the available scholarships and their eligibility for them. In contrast, the criteria for eligibility encompass a broad spectrum of factors; everything from academical achievement, to simply being born in a certain area. Simultaneously, there exists a large amount of Swedish scholarship providers where funds can go unclaimed.

1.2 Scholaro

Up until now, there has existed no centralized platform gathering scholarships in Sweden for easy access. Instead, individuals have been required to conduct manual searches online. This complicated scholarship matching processes is something Scholaro, a newly founded start-up based in Stockholm, aims to automatize. They have compiled an extensive database of approximately 500 Swedish scholarships. Utilizing generative artificial intelligence (GAI) and large language models (LLMs), Scholaro matches individuals' profiles with suitable scholarships, currently employing the large language model GPT-4 for this purpose. Scholaro does not have exact information on the cost for the GPT-4 usage as of today. However, the costs for computational resources for LLMs are increasing significantly due to several reasons, one being their expanding user base. Furthermore, Scholaro also seeks to extend their GAI usage beyond scholarship matching, opening up for other areas of use such as generating and submitting scholarship applications. Consequently, Scholaro is exploring the emerging field of small language models (SLMs). These more compact models are often available for free, open-source, requiring significantly less computational capacity, and processing power.

1.3 Problem formulation and research questions

The need for computational and financial resources is significant for LLMs, which creates an incentive of finding alternative approaches. A possible solution is employing smaller language models. A common approach to enhance the competitiveness of smaller language models against their larger counterparts in certain tasks involves fine-tuning them. However, this research is focused on examining and assessing models that have already undergone fine-tuning, including instruction-based models and chat models accessible through open-source platforms. This approach is adopted due to the significant amount of training data required for fine-tuning, which necessitates substantial resources for data collection.

In Sweden, a common form of student scholarship is awarded based on the merit of master's theses. These scholarships are retroactive, awarded based on a student's previous work, and provide financial support at the end of their studies. For such scholarships, applicants typically submit their thesis or its abstract, which then undergoes review by designated individuals to ascertain its relevance and suitability. For instance, scholarships focused on sustainability are exclusively available to students whose master's theses address sustainability issues. Therefore, the objective is to match master's thesis abstracts with corresponding scholarships efficiently. This process of scholarship matching neces-

sitates the application of three principal natural language processing (NLP) tasks; text classification, text summarization, and translation. Text classification facilitates the identification and recommendation of potential scholarship recipients according to the specific criteria of each scholarship. Text summarization is employed to compile concise summaries of scholarship applications, thereby enabling the creation of a database cataloging prior submissions by students. Translation is essential for converting applications between Swedish and English, depending on the languages preferred or required by the scholarship guidelines.

With this in mind, the aim is to explore the SLMs' capabilities of classifying and summarizing abstracts in English. Furthermore, their proficiency in translating between English and Swedish is examined, along with their understanding of the Swedish language. An investigation of their cost-effectiveness is also performed.

The research questions are:

- Which automatic and human evaluation metrics are most effective for assessing the performance of an SLM's ability to classify, summarize and translate texts from English to Swedish?
- In terms of these evaluation metrics, how do SLMs perform compared to each other and to GPT-4 when classifying, summarizing, and translating master's theses?
- Which evaluation metrics could be used to assess the SLMs' bilingual capability to comprehend Swedish prompts and generate Swedish text, and how well do the SLMs perform in this area compared to each other and to GPT-4?
- Apart from the automatic and human evaluation metrics, what other surrounding factors could be considered when choosing SLMs for these tasks? How can they be measured and how do the SLMs compare to each other and to GPT-4?

1.4 Related work

This section describes the prior work conducted in the fields of evaluating SLMs and Swedish comprehension of language models.

1.4.1 The performance of small language models

As an alternative to LLMs, SLMs have recently demonstrated their effectiveness regarding both computational resources, carbon footprint and performance. Firstly, the extensive

computational requirements of LLMs make them difficult to deploy in most applications used by organizations. In a recent study carried out by Google Research and the University of Washington it is stated that an LLM of the size 175 billion parameters requires 350GB GPU memory to serve [2]. Secondly, such large computational power results in a significant carbon footprint. Thirdly, studies have shown that SLMs can compare to the performance of LLMs, if they are trained on high-quality data. For instance, in a study carried out by Microsoft Research and Carnegie Mellon University, fine-tuning a model of parameter size 1.3 billion achieved over 81% accuracy on math questions [3]. Consequently, the SLM outperformed state-of-the-art LLMs tested on the same task.

Considering the performance of SLMs for natural language processing (NLP) tasks specifically, a study at the Center for Information and Language Processing at LMU in Munich showed that SLMs could perform similar to large models such as GPT-3 [4]. The results were achieved by pattern-exploiting-training, which merges the concept of rephrasing tasks into fill-in-the-blank questions with the standard process of fine-tuning. Furthermore, a study done by Microsoft Research demonstrated that a model with the size of 13 billion parameters performed comparably to models that are 5 to 10 times larger, after being fine-tuned to have better reasoning skills [5].

1.4.2 Evaluation of small language models

The evaluation of SLMs is to some extent inadequate and many evaluation protocols for SLMs are limited to a small number of tasks and different prompts. Moreover, the small models are often evaluated using larger model, such as GPT-4, judging their performance. However, recent studies have shown that this approach can be biased and is dependent on the order in which the generated responses are sent into the LLM [6].

However, for some of the recently launched SLMs, the developers publish papers comparing the model to competing models on some different benchmarks. Microsoft recently published one of the first extensive evaluations of a SLM. Their newly launched small model Orca-2 was tested on a total of 15 benchmarks, in approximately 100 different tasks [5]. In this evaluation, the benchmarks were chosen using suggestions from Open-LLM leaderboard and InstructEval [7] [8]. These suggestions have mainly been utilized for LLMs before, and a large-scale framework adapted for SLMs specifically has not yet been published.

1.4.3 Swedish in language models

The majority of today's LLMs are trained mostly on English data. The same goes for the SLMs; although some of them claim to be somewhat "multilingual", this often includes only widely spoken languages such as Spanish, French and German. In 2023 however, AI Sweden and RISE introduced GPT-SW3, the first native set of language models for the Nordic languages [9]. The main issue in building such models adapted for a smaller language such as Swedish, is to access sufficient amount of high-quality data. For GPT-SW3, a dataset called The Nordic Pile was made, containing text data from various open websites and forums in Swedish. Furthermore, GPT-SW3 distinguishes itself from most models since it is fully publicly available for research purposes. In the paper published in conjunction with the release of the model, it is mentioned that evaluation benchmarks for language models trained on other languages than English are to a large extent insufficient or non-existing. Therefore, English language evaluation benchmarks were used during the evaluation of GPT-SW3.

1.5 Contributions

In this section, the contribution the project aims to make to the research field of SLMs is introduced, encompassing both their evaluation and prompting. Additionally, the division of work among the authors is outlined.

1.5.1 Contribution to research

This thesis focuses on the evaluation of SLMs, developed between May 2023 and February 2024 and within a size range of around 7 billion parameters. The evaluation metrics extend beyond the mere performance of the models, as other aspects such as cost-effectiveness are also considered. Additionally, the models' understanding of Swedish instructions and the ability to translate text from English to Swedish are examined. Although the primary focus is the classification, summarization, and translation of master's thesis abstracts for scholarship matching, the evaluation framework presented in this paper is intended for other purposes across various domains where SLMs can be applicable.

The research further includes exploring several known prompting techniques, that have previously been applied to LLMs, but not to SLMs to the same extent. Based on this research, the most effective prompt for categorizing thesis abstracts amongst a list of subjects is identified.

Furthermore, the dataset, consisting of 300 master thesis abstracts, labeled according to main subjects, alongside English summaries and Swedish translations, is available open source and can be used to perform further research within the field.

1.5.2 Division of work between authors

In the beginning of the project, we collaboratively explored and selected appropriate language models, as well as conducted the initial NLP tests. For the division of conducting research for the background section, Amanda covered the parts about artificial neural networks and transformers, while Allis focused on the advantages and limitations of LLMs and SLMs. Regarding creating the dataset, we evenly split the task of classifying and labeling. As the project progressed, Amanda focused on prompt engineering, doing research on various prompting methods and simultaneously evaluating them. Meanwhile, Allis primarily focused on identifying suitable evaluation metrics as well as implementing them in the code. The final testing phase was performed together, in order to guarantee the reliability of the results.

1.6 Limitations

In the process of selecting SLMs for evaluation in this project, certain distinctions were established. Initially, a comprehensive research was made to identify the best, most recent, and widely used models suited for similar tasks. During the preliminary phase, approximately 15 models were chosen for initial testing, primarily to assess their performance on simple classification tasks. A subset of these models was subsequently excluded, on the basis of three different criteria. Firstly, some models were too specialized for specific tasks, whereas the requirement was a versatile model capable of performing a variety of tasks. Secondly, the availability of models was a consideration; for some, only the base versions were accessible open-source, lacking instruction-tuned versions, thus necessitating fine-tuning. Thirdly, the exclusion was also based on deficient performance in the initial evaluations. After refining the prompts and conducting extensive tests, several models failed to achieve a sufficient F1-score leading to their elimination from the evaluation set. The following models were excluded based on the criteria mentioned:

Specified only for classification (one task): BERT (by Google) and RoBERTa (by Meta).

No instruction-tuned version available: Phi-2 (by Microsoft)

Deficient performance: Tinyllama (by Meta), StableLM (by StabilityAI), Llama-2 (by Meta), Mpt-7b (by Mosaicml), Falcon-7b (by TII), Gemma-7b (by Google), and Olmo-7b (by Allen AI).

Note: the initial tests were conducted in March 2024. Since the evaluation of language models is a fast moving field, some of these models might have instruction-tuned versions available open-source or exhibit efficient performance for our tasks in the future.

1.7 Outline of report

The report begins with a theoretical background in Chapter 2, describing the foundation of machine learning, artificial neural networks, and transformer models. It also reviews the fundamentals, applications, and limitations of both large and small language models. Additionally, the various evaluation metrics used to assess the language models in this project are presented. Chapter 3 outlines the methods used to gather the test dataset, create the prompts, and evaluate the language models. The results are provided in Chapter 4 and further discussed in Chapter 5, which also suggests directions for further research. Finally, the research questions are answered in the conclusion in Chapter 6.

Chapter 2

Background

This chapter describes the underlying theory of machine learning, artificial neural networks, and transformers, which form the basis of the evaluated models. Additionally, the difference between large and small language models are discussed, as well as the significance of prompting correctly. Lastly, some important automatic model evaluation metrics are described.

2.1 Machine learning

Machine learning refers to computer programs that are able to perform tasks by learning from data the program is trained on. It mainly consists of three parts; the training data, a mathematical model that represents the data, and a learning algorithm. The learning algorithm is able to automatically adjust the parameters of the mathematical model to agree with the data [10].

2.2 Artificial neural networks

Artificial neural networks (ANN) are algorithms that are inspired by the human brain. Similar to the brain, the network acquires knowledge from a learning process and uses neurons, or weights, to store the acquired knowledge. ANNs are able to perform several different tasks, including regression, classification and sequence prediction [11].

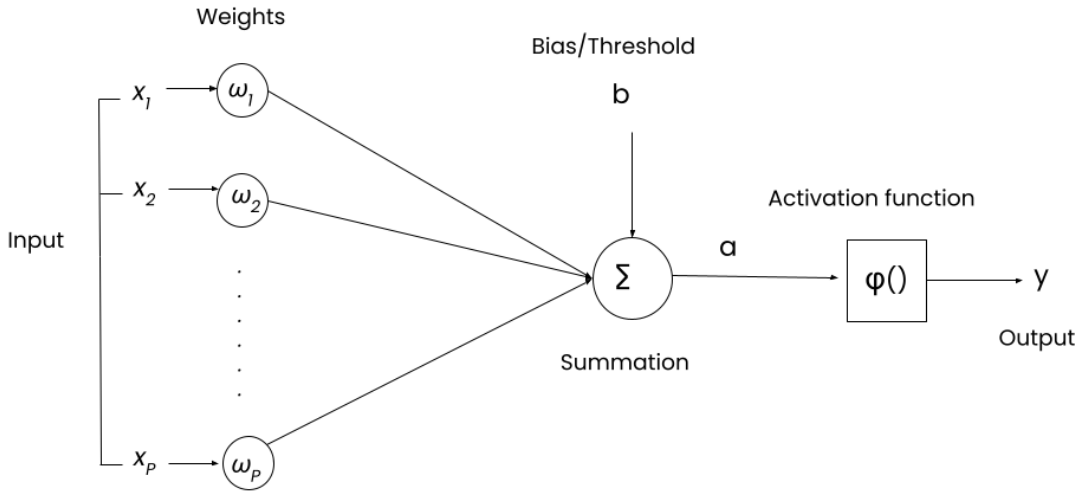


Figure 2.1: The basic element of an ANN. Figure inspired by [11].

2.2.1 Loss function

The way of measuring the performance of an ANN depends on which specific task the network is supposed to perform. During training, the loss function, or the error between the predicted and the given value, is often used as a measurement. The goal is naturally to minimize this loss. The minimization is performed during training by using a specific optimization method. The most common optimization methods are *gradient descent* and *stochastic gradient descent* [11].

2.2.2 Network architecture

The basic element of an ANN is shown in Figure 2.1. A weighted summation of the input signal outputs a new signal that is transformed with an activation function, $\varphi(a)$, which in turn generates an output. The type of activation function is often chosen depending on what task the network is meant to perform. Common for most activation functions is that they give a larger output for an increased input and have threshold levels for some critical point. The weighted summation a is calculated in a computational node as

$$a = \sum_{k=1}^{K} \omega_k x_k + b,$$

with x_k being the inputs shown in Figure 2.1 and K the number of inputs. The weights, ω_k, are parameters that can be tuned during training of the network, to yield the best output. Thus, the input to the activation function is also tuned when the weights are updated. By performing this tuning, features of higher importance for the final output can be given increased weight, and vice versa. Additionally, a bias term b is added to offset the value of the weighted sum. The bias is also learnt during training as an additional parameter, and can be compared to the y-intercept in the line equation. The bias term additionally provides higher probability for a correct output. The number of weight parameters is essentially what is referred to as the *size* of the neural network [11].

The manner in which the computational nodes are connected is referred to as the network's architecture. The simplest architecture is the so-called perceptron seen in Figure 2.1, which consists of only one node in one layer. However, the nodes can lay in several hidden layers; an architecture that is referred to as deep learning, as shown in Figure 2.2. The input layer consists of n nodes, which outputs go through m hidden layers of nodes (h_i), resulting in k outputs. It is preferable to have different number of nodes in each layer. Each node in every layer receives inputs from the previous layer, processes these inputs using a weighted sum followed by an activation function, as for the single perceptron, and passes the output to the next layer. The activation functions must be non-linear, since the network otherwise can be represented by just the simple perceptron [11].

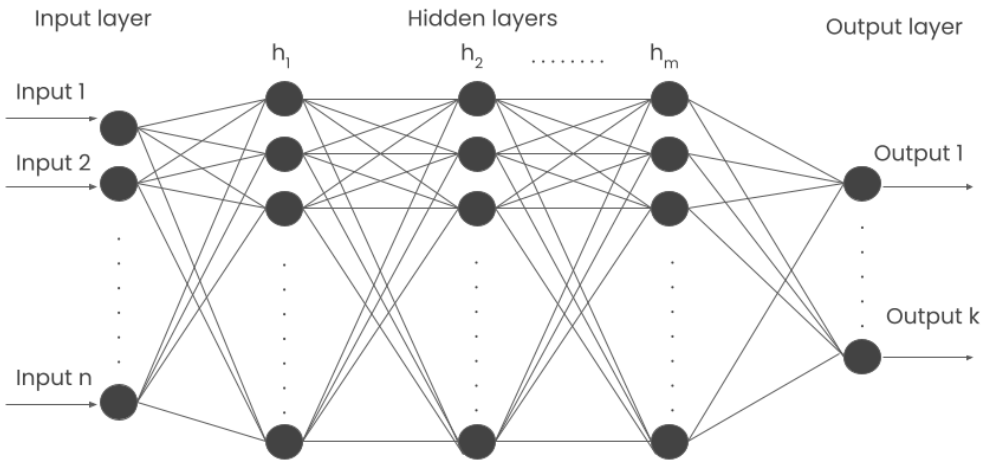


Figure 2.2: An ANN with m hidden layers (deep learning architecture). Figure inspired by [12].

2.2.3 Activation function

The choice of activation function depends on the task. Moreover, the activation functions can vary in different layers of the network. For classification tasks, a commonly used activation function in the output layer is the *sigmoid* function, which maps the input to a number between 0 and 1. The main problem with the sigmoid function is that it can contribute to vanishing gradients when input values are too small, which makes the network less sensitive to changes and slow to train. The rectified linear unit function, or the *ReLu* function, solves this problem by mapping all negative values to zero. This activation function is usually used in the middle layers of a deep learning model, since it can decrease the computation time. For multi-label classification, the *softmax* function is popular. In the softmax function the outputs are normalized, creating an output vector that represents a probability distribution over the likelihood of different classes [11].

2.2.4 Feed-forward and recurrent networks

The two major types of ANN architectures are feed-forward networks and recurrent networks. Feed-forward networks are restricted to the fact that input values propagate straight through to the output layer. On the contrary, in recurrent networks there are feedback loops that allow information to persist, resulting in no distinguished inputs or outputs [11].

2.3 Transformer models

Transformer models have been widely used for NLP tasks since the proposal in the paper "Attention Is All You Need" published by Vaswani et al. in 2017 [13]. Before, the state-of-the-art models were the recurrent neural networks (RNN) mentioned in Section 2.2.4, and long-short-term-memory models (LSTM). LSTMs were introduced to solve the problem with the vanishing gradient in RNNs. This vanishing gradient resulted in that earlier inputs decreased exponentially, which made the model miss out on long-term dependencies [13].

The transformer models have been revolutionary for NLP tasks since they, due to their self-attention layers, can handle larger input sequences of text. Furthermore, they are also able to analyze words in parallel, a great advantage for texts where the meaning of words depends on context. Moreover, there exist no recurrent units in the transformer models, which results in shorter training times [14].

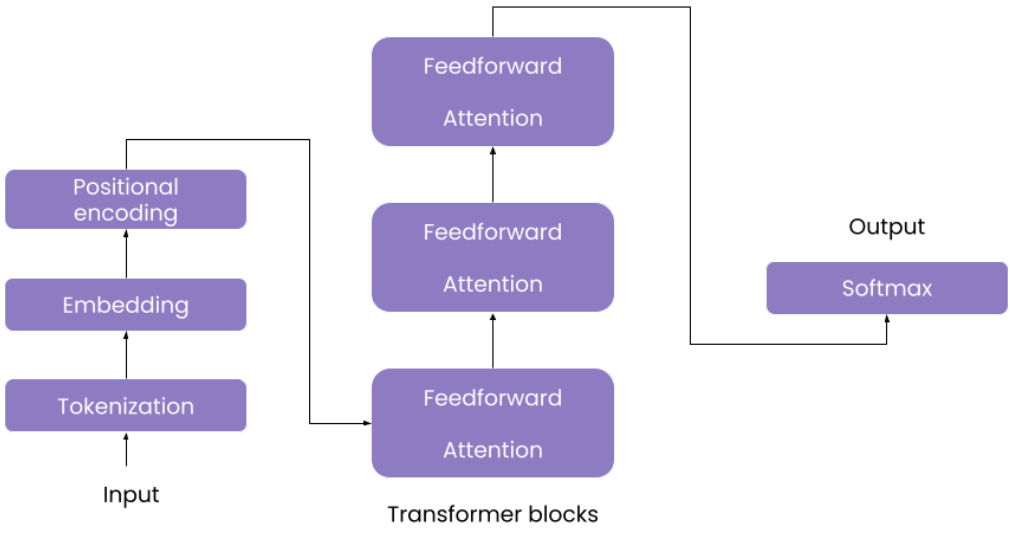


Figure 2.3: The main parts of the transformer architecture. Figure inspired by [15].

The main blocks of the transformer architecture are illustrated in Figure 2.3. In order to be able to fully understand how they work, the concepts will be introduced in detail in the following sections.

2.3.1 Tokenization

The initial phase of every transformer model includes a tokenization process, since the model necessitates preliminary processing upon receiving textual data for an NLP task. The tokenization process involves segmenting the text into individual tokens, which for instance includes words or characters. Such tokenization is essential for the model to manage linguistic variability and comprehend different meanings of identical words. Various tokenization techniques exist, and the choice among them depends upon the specific task the model aims to accomplish. For NLP tasks undertaken by transformers, a common method of tokenization is at the sub-word level. This approach aims to capture not only the significance of individual words but also the connections between sub-words within those words. It involves incorporating commonly occurring words into a vocabulary and divide less common words into prevalent sub-words.

For example, a word such as "annoyingly" might be tokenized into more frequently encountered sub-words [16]. An example could be:

"annoyingly" $\longrightarrow$ "annoying" and "ly"

2.3.2 Word embedding

Word embedding refers to the technique of converting words into numerical representations, enabling computers to read and process the input effectively. This process can be mathematically described as a parameterized function f_θ defined as

$$f_\theta(W) = \theta,$$

where θ is the parameter and W is the word in a sentence.

One embedding technique is the *one hot encoding*. While easy and fast to implement, it is not widely used in NLP since the context of the word is lost. One hot encoding builds upon converting every word in the vocabulary into an index in a vector. Consequently, a word is represented as a large vector with 0:s everywhere except for a 1 in the position which refers to that specific word. Hence, the size of the vector corresponds to the total number of words in the vocabulary. The one hot encoding technique is not able to understand the meaning of a certain word, or distinguish between the same word in different contexts [17].

Word embedding used in transformers deviates from one hot encoding since it apprehends the meaning of linguistic data. The semantic connection between different words is represented by the distinction between them in an n-dimensional space. For example, the words "Pasta", "Dog" and "Potato" are all distinguished. However, "Pasta" and "Potato" will be represented closer in the vector space than "Pasta" and "Dog". This is illustrated in a hypothetical two dimensional space in Figure 2.4. Word embedding approaches used in transformers are more dense and therefore often obtain dimensions lower than the vocabulary size [17].

Cosine similarity is a way to measure the distance between two vectors in an n-dimensional space, in order to determine the semantic similarity between the words. Mathematically, it is defined as the dot product between to words, divided by the Euclidean distance between the words. The cosine similarity, $cos(\theta)$, between two word vectors **A** and **B** is defined as

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}.$$

Here, θ is the angle between the vectors. A value of 1 indicates two identical words [18].

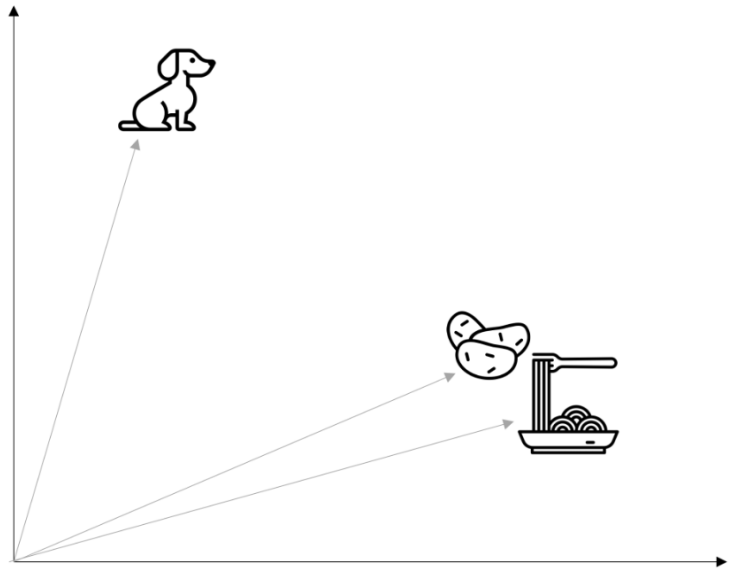


Figure 2.4: An illustration of how the words "Dog", "Potato" and "Pasta" could be represented in a two dimensional word embedding vector space.

Transformer models are using something called *learned word embedding*, which refers to an unsupervised learning method to understand semantic correlations between words [19]. The word embedding is found in hidden layers within the network, with weights that are being updated as the network is trained. The embedding works in a word-to-vector manner, where directions in a high dimensional space carry semantic meaning. Moreover, the learned embedding in transformer models are contextual, meaning that the same word will get a different embedding depending on the context it is found in [20]. Two commonly used word embedding methods are Word2vec and GloVe [21]. The learned embedding works similar to these methods, but each transformer model has their own weights, depending on their input training data.

2.3.3 Positional encoding

In the positional encoding a sequence of predefined vectors is added to the embedding vectors of the words, in order to get a unique vector for every sentence in the input text. A relative position for each word in a sequence, dependent on the words around that word, is then given in the form of a matrix representing all the positions. In order to generate a unique vector for each position, the sine and cosine functions are used. If the embedding vectors have the length d_{model} and the maximum number of positions in the embedding is L, the positional encoding $PE_{(k,j)}$ is given by

$$PE_{(k,2i)} = \sin\left(\frac{k}{n^{2i\,/\,d_{model}}}\right)$$
$$PE_{(k,2i+1)} = \cos\left(\frac{k}{n^{2i\,/\,d_{model}}}\right),$$

where the position goes from $k = 0$ to $L - 1$ and the position in the embedding vector from $i = 0$ to $d_{model}\,/\,2$. The value n can be set to any number. The encoding vectors can then be summed, since they all share the same dimension, i.e. d_{model} [13].

2.3.4 The transformer block

The transformer block consists of multi-head attention and a feed-forward mechanism. The attention mechanism works by contextualizing each token that is sent into the model with other tokens within the same context. The contextualization happens in every layer of nodes, via a parallel multi-head attention. This results in signals from tokens of importance being amplified, while signals from less important tokens will be reduced. Essentially, the attention mechanism is moving words in a piece of text closer to each other in the word embedding space. The transformer outputs a score for each word, where higher scores correspond to words that are more likely to follow next in the sentence [22].

The transformer block can consist of either an encoder, a decoder, or both an encoder-decoder architecture. The encoder functions by processing a sequence of tokens to produce a fixed-size representation of that sequence. Conversely, the decoder takes a fixed-size vector as input and uses it to produce a sequence of words. Encoder-decoder transformers are for instance used in language translation tasks where a sentence in one language is required to be processed in order to produce a corresponding sentence in another language. The encoder transformer is typically employed for classification tasks, as it is designed to accept a sequence of text and classify it into a predefined category. The decoder transformer is used for language generation, enabling the model to create text based on a given context or prompt. GPT-4 is an example of a decoder model [22].

2.3.5 Output layer

The output layer consists of the softmax activation function, which turns the scores from the transformer into probabilities. The outputs from the softmax are normalized, creating an output vector that represents a probability distribution. The standard unit softmax function, $\sigma(\mathbf{z})_i$, is defined by

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, ..., K \text{ and } \mathbf{z} = (z_1, ..., z_K).$$

The softmax function is applying the exponential function to each element of the input vector $\mathbf{z}$ and normalizes these values [23].

It is desirable to only let the word at a certain position depend on the outputs from earlier positions. Thus, in the output layer, it is ensured that all of the weights from subsequent positions are set to zero, not letting them affect the current position. This process is called masking. Mathematically, it begins before applying the softmax function by setting all of the future weights to negative infinity. The weights will then take the value 0 after being normalized by the softmax function [13].

2.4 Large language models

Large language models (LLMs) are a type of ANN that use the concept of deep learning. They are built on transformer architecture and are also the most common type of transformer model. The models build sentences by handling a classification problem, where they attempt to predict the probability of the next word in the sequence. The possible classes are all words that exist in the model's vocabulary [24]. The models are referred to as large because of the immense datasets they are trained on, and since the number of trainable parameters can reach up to 530 billion as of today's state-of-the-art models. The large datasets and number of parameters make the training phase of the models extremely computationally heavy [25].

By using unsupervised learning, meaning they are trained on unlabeled data and able to learn from huge unstructured chunks of data, LLMs have grown extremely popular during the last few years. However, the theory behind LLMs have been around since the 1950's.

Already in 1966, J. Eliza at MIT developed a simple program that mimicked natural human language by predicting the probability of the following word [26]. As computational resources developed to be faster and stronger, it eventually became possible to train the LLMs seen today.

2.4.1 Applications and limitations

Today, LLMs converse fluently and are able to assist humans with various sorts of tasks, such as generating and summarizing text, calculating, coding and analyzing sentiments. They have become experts on NLP; the technology on interpreting, generating and understanding human language. Since the models are unsupervised, they are somewhat self-learning as they acquire more data. Therefore, they can to a certain extent, answer any type of query and make the response make sense without any limitations to the number of instructions or questions.

However, a limitation of LLM is, while they have access to enormous amounts of data, they are also limited to that data. Firstly, if the training data is false, the model will also produce false answers. Secondly, even if the training data is correct, the model might hallucinate inaccurate answers. Hallucinations happen if the output is incorrectly decoded by the transformer and can be compared to the way humans misinterpret patterns that are not actually there [27].

2.4.2 Implementation costs

To operate an LLM like OpenAI's GPT-4, expenses are generated for each prompt token or for every sampled token. As of the current date, GPT-4's pricing varies by the engine used and its maximum context length. The maximum context length denotes the sum of the prompt length and the maximum token count for completions. For the standard model with a context length of 8,000 tokens, the rate is $30 per 1 million prompt tokens or $60 per 1 million sampled tokens. With a context length of 32,000 tokens, these rates rise to $60 and $120, respectively [28].

2.4.3 Environmental impact

The environmental impact of LLMs is usually an overlooked metric, as other performance metrics, such as accuracy and effectiveness, are more common in the initial implementing phase. One reason for this might be that a standardized way to measure the amount of carbon emission released when training a machine learning model is yet to be developed

[29]. This makes the environmental impact difficult to compare between different models. However, the environmental impact is generally quantified by a model's carbon footprint. Carbon footprint measures the amount of greenhouse gas emissions, usually in tonnes of CO2 equivalents, released in a certain process.

Some developers are open with the amount of carbon emission they release, while others are not or do not have the resources to estimate it. Nevertheless, it is clear that LLMs require a substantial amount of computational resources, and thereby energy, during their training phases. Thus, the carbon footprint of a model depends on the amount of computational resources needed, how much energy the computation requires, and where the energy is sourced from. The division of energy sources vary significantly depending on location and the choices the developers make when deciding on energy supply. Those choices can dramatically influence the model's overall carbon footprint, potentially varying by a factor of up to 60, according to a study from 2023 by Alexandra Sasha Luccioni et al. [30].

Moreover, the energy consumption for inference with the LLMs is also an important factor. A study conducted in 2023 by Siddharth Samsi et al. analyzes the energy consumption for inference of different sizes of Meta's LLama models, on NVIDIA's GPUs V100 and A100. The study concludes that the smaller Llama-models (with 7 billion parameters) consumes about half as much energy per second as the largest Llama-model (with 65 billion parameters). Factors such as batch size and number of GPUs used affect energy consumption per second and energy consumption per decoded token [31].

In an attempt to quantify the carbon emission from an LLM, David Patterson et al. estimate that 552.1 tonnes of CO2 eq were used during the training phase of GPT-3 (with 175 billion parameters) [32]. In a paper from 2022, Alexandra Sasha Luccioni et al. estimate the carbon footprint of an LLM of the same parameter size, BLOOM (with 176 billion parameters), to 24.7 tonnes of CO2 eq [29]. These two estimations give an indication of how large the carbon footprint of training and deploying LLMs can be.

2.5 Small language models

Since LLMs require a lot of time, resources, and power for training and maintenance, as well as substantial data storage, SLMs have become a popular alternative. These models are, as the name suggests, trained on smaller datasets and contain fewer parameters. The exact limit between a large and a small language model is not widely acknowledged, but it is safe to state that a model with 1 billion parameters is considered small, while models with 180, 70, or even 40 billion trainable parameters are considered large [33].

2.5.1 Why SLM instead of LLM

The advantages of SLMs are many, including less computational costs and less data for training. As the smaller models require less computations, the code for implementing the models can sometimes be run on local servers. This makes the data more safely stored compared to using an LLM which usually requires cloud based storage. Additionally, handling smaller datasets automatically makes it easier to control the fairness and bias of the data, thereby strengthening the security. Another benefit with SLMs is the fact that they are usually faster to deploy compared to LLMs [34] [35].

SLMs can handle a lot of the same tasks as their larger counterparts. However, it has been shown that a model's performance varies with its size. This means that the SLMs are not performing as well overall, e.g. with several tasks such as mathematics, natural language generation, summarizing, coding etc. However, it does not necessarily mean they will not outperform LLMs in certain, specific tasks. On the contrary, SLMs that are fine-tuned on specific datasets have, for some particular tasks, yielded more accurate results than larger models [35].

2.6 Prompt engineering

Prompt engineering refers to designing and refining the prompt to achieve the most efficient and precise output. It has been shown that different prompt designs have a significant influence on the output generated by language models. If prompt engineering is done in a proper way, the need for fine-tuning with new data can even in some cases be avoided. Research in the area of prompt engineering for LLMs, such as GPT-3.5 and GPT-4, has been extensively conducted over the recent years. For instance, it has been proven that the more precise a prompt given to an LLM is, the more accurate the response will be [36]. A prompt benchmark called *PromptBench* shows that LLMs are sensitive to adversarial prompts, or contradictory instructions, where careful prompt engineering has been shown

to affect the performance of the model [37]. However, not much research on how SLMs respond to differences in the prompt has been published.

The prompting techniques used in this project are presented in the following sections. These techniques were selected because they were the only ones identified within the project's time frame, ensuring that no other techniques were overlooked. However, it is possible that other techniques exist, and new ones are continually being developed. The techniques presented here were determined to be the most well-researched for interacting with LLMs.

2.6.1 COSTAR prompting framework

COSTAR is a prompting framework, proven to work well with LLMs. COSTAR stands for context, objective, style, tone, audience, and response. In the "context", background information is given in order to improve the model's understanding of the scenario. The "objective" should clearly state the model's task. The "style" and "tone" ensure the correct sentiment of the response and the "audience" specifies the targeted user. Lastly, giving the format which the "response" should be given as, produces the wanted output [38].

2.6.2 Prompting SLMs using LLMs

In the community of SLMs, a newly introduced technique for prompting the SLMs letting the LLM GPT-4 write a suitable prompt, which has proven to be effective [39].

2.6.3 Few-shot learning

One way to improve the performance of a language model is to apply one- or few-shot learning in the prompt. This is a type of prompt based fine-tuning, where the model learns from one or a few examples of labeled data. In a recent study, conducted by Microsoft Research, experiments showed that larger models benefit more from few-shot learning than small models do [5].

2.6.4 Chain-of-thought prompting

A technique proven to improve both zero- and few-shot ability for LLMs performing reasoning tasks, is the Chain-of-thought (CoT) prompting. This technique includes adding a sequence of intermediate reasoning steps in natural language into the prompt when assigning a task to the model [40].

2.7 Model evaluation

In order to assess the performance of a language model, evaluation is crucial. There are several different evaluation methods, and depending on the task and application different methods are suitable [41]. Here, some of the most established metrics are presented. The chosen evaluation metrics included in the framework were selected as the most suitable options identified during the project's time frame. When creating a suitable framework for the evaluation of language models, it is important to ensure that each metric contributes with a unique perspective to the evaluation. Including too many similar metrics does not add significant value to the evaluation.

For the classification problem, other metrics, such as precision, recall, and the ROUGE-L score, were excluded from being presented in the results, due to their similarity to the chosen metrics. Details on these three metrics are further explained in Section 2.7.1. The additional metric AUC-ROC curve was excluded because it is not primarily suitable for multi-label classification, which is the focus of this project. While it is possible to illustrate several classes in the ROC curve, it becomes impractical with a large number of classes, such as the nearly 60 classes presented in this project. This metric is further explained in the end of Section 2.7.1. For the evaluation of the summarization and translation tasks, the selected metrics were those identified as sufficiently informative within the project's time frame. Metrics such as METEOR, NIST, LEPOR, and WER were excluded because ROUGE, BLEU, and BERT-F1 were considered to provide adequate information within the project's duration. These excluded evaluation metrics are further explained in the end of Section 2.7.1.

The combination of metrics presented in the final framework, found in Section 3.5, ensures a thorough and multifaceted evaluation, as they all bring unique strengths that collectively offer a robust understanding of a model's performance. Furthermore, the strengths and weaknesses of the chosen evaluation metrics for this project are discussed in Section 5.2.

2.7.1 Automatic evaluation

Automatic evaluations are a common way to assess a model's performance. There are different metrics tailored for specific tasks; for NLP tasks such as classification, but also for knowledge in areas such as coding and mathematics. The main benefit of automatic evaluation is that it is standardized, and therefore suitable when comparing models to each other. Furthermore, automatic evaluation metrics are easy and quick to use, leave minimal room for human error and no extensive human participation is required [42].

F1-score

The F1-score is one of the most commonly used validation points for evaluating classification tasks. It combines the metrics precision and recall. Precision measures the proportion of instances the model classifies correctly for each class. Recall indicates the proportion of how many instances of one class that were found and correctly predicted to that class. Both precision and recall are calculated by counting the so called True Positives (TP), False Positives (FP), and False Negatives (FN). For one specific class, say for example biology, the terms refer to:

- TP: The model correctly classified a text about biology as biology.
- FP: The model incorrectly classified a text that was not about biology as biology.
- FN: The model incorrectly classified a text that was about biology as something else.

Precision and recall are defined as

$$\text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Precision and recall balance each other out by being combined into the metric F1-score, ranging from 0 to 1. An F1-score of 1 indicates perfect precision and recall, while a score of 0 indicates no matching at all between predicted and actual values. Which F1-score can be considered good is different for different applications, but to reach a score of 1 is very difficult in practice. The F1-score is defined as [43]

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

ROUGE score

The ROUGE score is widely used for evaluating summarization tasks and has been shown to correspond well to human evaluation [44]. It involves comparing a model's output with a reference summary, written either by a human or by an LLM [45].

ROUGE-1, ROUGE-2 and ROUGE-L are different types of ROUGE scores. ROUGE-1 is comparing the overlap of unigrams (separate words) between the model's output and the reference summary, breaking down all sentences into their individual words for comparison. ROUGE-1 is considered excellent at a score of 0.5, and moderate around 0.4. ROUGE-2 works in a similar manner by instead comparing bigrams, or pairs of words followed by each other. For instance, in the sentence "Small language models rock", there are four unigrams (small, language, models, rock) and three bigrams (small language, language models, models rock). The ROUGE-2 approach offer more insight into the flow of the summary. A third version of ROUGE score is the ROUGE-L score, that assesses the length of the longest common sub-sequence shared between the summaries. For ROUGE-2 and ROUGE-L, a score of 0.4 is considered good [45].

BLEU score

The BLEU score is a measurement of a model's capability to translate, proposed by Kishore Papineni et al. in a paper from 2002 [46]. The algorithm initially compares the n-grams in the model's output to the n-grams in a reference translation, in a similar way as the ROUGE score. Hence, the BLEU-2 score measures the bigrams in a similar way as ROUGE-2 score does. However, the difference is that the BLEU score does not take the order of the tokens in the bigrams into account. For example, the BLEU score does not differ between the phrases "Master thesis" and "Thesis master". BLEU also adjusts the score by a brevity penalty to punish overly short translation candidates. A BLEU score of 1, where the candidate text is identical to the reference, is extremely rare to achieve in practice due to the complexity and variability in human languages. Consequently, scoring over 0.3 is considered good. The BLEU score does not take synonyms into account, which is a common issue for n-grams-based evaluation metrics, and can thereby give misleading results [46].

Perplexity

Perplexity, $PP(p(x))$, is a metric determining how well a model can predict a sample, e.g the following generated word. Perplexity can be defined as the normalized inverse uncertainty a model has in predicting the next word in a sequence, as follows

$$PP(p(x)) = 2^{H(p(x))}.$$

Here, $p(x)$ describes the probability distribution function of the model's training data and x are the tokens or words. In other words, for every token x, $p(x)$ describes the probability distribution. $H(p)$ is the entropy of the probability distribution $p(x)$ and is defined as

$$H(p) = -\sum_{x} p(x) log_2(p(x)).$$

Intuitively, perplexed is a synonym to surprised, and therefore the perplexity score measures how surprised the model is of encountering new words. A low perplexity score indicates a more accurate model with better understanding of the language. However, it is difficult to state whether a score is considered good or not, since it depends on how well other models perform. Hence, the perplexity score can be considered more of a comparison tool between different models [47]. The average perplexity of GPT-4 is 2.6 and for GPT-3.5 it is 4.5. Thus, every score within a range somewhat close to those values can be considered very good [48].

Perplexity is a metric that completes the evaluation of language models since it does not evaluate the model's performance on a certain task, but rather the model itself. In contrast to metrics such as the F1-score, the ROUGE score, and the BLEU score, it evaluates the model's capability to naturally generate a fluent and coherent text [49].

BERT-F1 score

The BERT-F1 score is a metric that compares a candidate text to a reference text with cosine similarity. In contrast to ROUGE and BLEU, which also utilize a reference text, BERT-F1 takes synonyms into consideration by calculating the cosine similarity between two words in the way described in Section 2.3.2. The BERT-F1 score is supported by several languages, including Swedish. The output gives the model's precision, recall, and F1-score [50].

AUC-ROC curve

The ROC curve is a plot of the true positive rate against the false negative rate, as shown in Figure 2.5. The true positive and false negative rates are described in Section F1-score. The ROC curve typically illustrates the performance of a binary classifier at various threshold settings. Given a threshold parameter, an instance is classified as positive if the score computed for that instance is greater than the threshold, and as negative if the score is smaller. The AUC, or area under the curve, represents the probability that the model

will rank a randomly chosen positive instance higher than a randomly chosen negative instance. A perfect classifier would reach the top-left corner in Figure 2.5, while a random guess would result in a diagonal line, corresponding to an AUC of 0.5 [51].

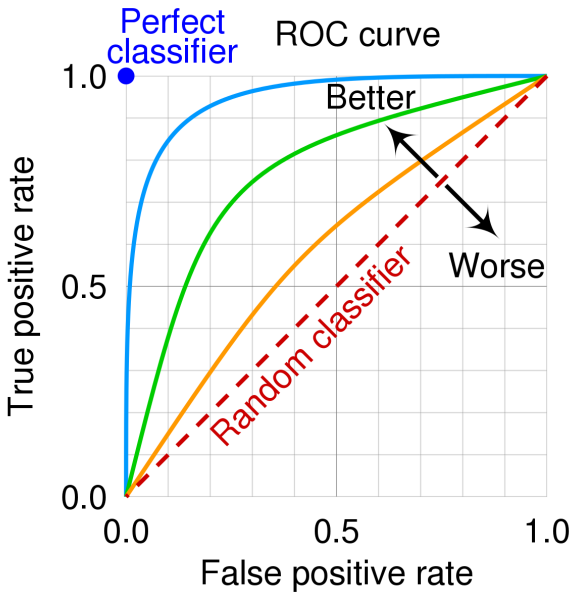


Figure 2.5: The ROC curve [51].

METEOR score

The METEOR score is a metric developed to evaluate machine translation. Similar to BLEU, it maps unigrams in a translated sentence to those in a reference sentence. However, METEOR considers both precision and recall, weighing recall higher. The METEOR algorithm performs exact word matches as well as matches based on identical stems and synonyms when comparing words in a translated text to those in a reference translation. METEOR can, similar to BLEU, also be used to evaluate summaries if a reference summary is provided [52].

NIST score

The NIST evaluation metric is an adaptation of BLEU, and works similar by comparing a machine translation or summarization to a reference translation och summary using n-grams precision. The difference is that NIST gives more importance to less frequent n-grams, since the information gain from each n-gram is taken into account [53].

LEPOR score

The LEPOR score is used to evaluate machine translation and summarization, and works similar to the BLEU score. It is specifically designed to address language bias, where some evaluation metrics may perform inconsistently across different languages, by incorporating tunable parameters to mitigate this issue. Additionally, the LEPOR score includes an enhanced length penalty, which penalizes translations that are significantly longer or shorter than the reference translation. It also features an n-gram word order penalty, which penalizes translations with n-gram sequences that differ in order from those in the reference translations [54].

WER score

WER, or Word to error rate, is an evaluation metric used to assess machine translation and speech recognition. It measrures the ratio of errors in a machine translation to the total words in a reference translation. It is calculated as

$$WER = \frac{S + D + I}{N},$$

where S is the number of substitutions (words that are incorrectly translated compared to the reference translation), D is the number of deletions (words that are missing compared to the reference translation), I is the number of insertions (extra words that are included in the machine translation and not in the reference translation), and N is the total number of words in the reference t. However, WER is mainly focused on the accuracy of transcribing speech to text, rather than translating from one language to another [55].

2.7.2 Human evaluation

Human evaluation is important in order to produce a balanced overall evaluation of a language model. Moreover, for some NLP tasks the automatic evaluation metrics available today are not suitable, why human evaluation is necessary. Examples of such tasks are open-generation tasks, where embedded similarity metrics are not enough for a full assessment. Moreover, for some applications manual evaluation can align closer to the actual experience for a user, why humans in those cases can provide more accurate feedback [41].

Chapter 3

Method

This chapter introduces the methodology used to create the dataset, select models and suitable prompt, as well as the coding details. Furthermore, how the evaluation metrics were applied and the way the models' cost-effectiveness was measured is described.

3.1 The dataset

The dataset methodology was divided into three different stages; the gathering of the abstracts, the pre-processing of the texts, and the labeling, including both manual and GPT-4 generated.

3.1.1 Data gathering

A dataset consisting of 300 master thesis abstracts, sourced from the Swedish digital portal DiVA (Digital Scientific Archive), was compiled. DiVA archives a collection of close to 70 000 master's theses written by students from Swedish universities in various academic fields. The goal was to assemble a dataset that to some extent would reflect the broad spectrum of subjects covered in these master's theses. According to statistics from DiVA, the ten most common subjects across the collection of master's theses were identified to be the following: [56]

- Technology (25%)
- Social science (18%)
- Natural science (16%)
- Computer science and information technology (8%)
- Humanities and arts (7%)
- Economy and business (6%)
- Medicine and health studies (5%)
- Mechanical engineering (5%)
- Business administration (4%)

Note that these subjects are not the same as the labels that were used for classification. The labels used for the classification task consisted of 59 different and more specific subjects, and can be found in Appendix A. The collection of the data was adapted according to the distribution of subjects given in the list above. It was assumed that DiVA reflects the spread of different subjects in Sweden well, since they collect papers from 50 different universities and research institutions across the country.

Given that the DiVA website offers open access to master thesis abstracts, it is authorized to use it for research purposes [57].

3.1.2 Preprocessing of dataset

As for the preprocessing of the collected dataset, all new lines, any titles, and other unnecessary information were removed. Any keywords included in the abstracts were eliminated as well, as they might be interpreted as correct labels for the classification task. Additionally, other formatting issues were resolved.

Furthermore, all abstracts were divided into six different documents. The distribution of abstracts across different academic fields remained consistent with the same spread described in Section 3.1.1 for all six documents. However, in order to keep the same spread of academic fields within the six documents, they had to contain 47 abstracts each instead of 50. As a result, only 282 abstracts from the original dataset were used as test data. The reason for dividing the dataset into smaller sections was to minimize the time required for testing different prompts.

For the classification tasks, all of these six documents were taken into account and the average performances of the models were calculated. However, the extended duration required to run the models on the summarization and translation tasks made the testing of all 282 abstracts impractical. Instead, for the summarization task only one of these documents, i.e. 47 abstracts, was used. For the translation task, we halved this number and only 25 abstracts were tested, since this task required even more time. The same 47 or 25 abstracts were tested on all of the models, to receive as fair results as possible between the models.

3.1.3 Labeling of dataset

In order to be able to evaluate how well a model performs in a certain task, the dataset needs to be labeled. These labels serve as a benchmark against which the models' outputs can be compared. Moreover, the labels can be used for one- or few-shot learning techniques during the prompting process. These techniques are further discussed in Section 3.3.4. The approach to label the test dataset depended on the task, and all of label techniques are described in the sections below. Common for all of the labeling approaches was the utilization of Open AI's GPT-4 model, to either make the labels or to complete the already existing labels. This was done in order to be able to benchmark the models' performances against GPT-4. As of May 2024, GPT-4 is one of the most efficient and established LLMs available. Notably, it is also the model currently used by Scholaro for their scholarship matching, making it particularly interesting for our research.

The 300 abstracts with classification and summarization labels, and 50 abstracts with translation labels can be found in Appendix A.

Classification task

The abstracts were categorized under various labels including technology, artificial intelligence, sustainability, computer science, psychology, among others. The most suitable labels were chosen with influence from DiVA's categorization of the papers and the authors' tagged keywords, but mainly from our judgment as human readers. This task showed to be challenging due to the strong overlap among different academic fields. Furthermore, the question of the specificity of the labels arose. A weigh-off between the possibility of assigning increasingly specific labels to an abstract and the need of maintaining a limited number of labels within the dataset presented itself. Consequently, each abstract was limited to a maximum of five labels, and some specific categories were clumped into a single broader category. For instance, the subjects 'statistics', 'probability theory', 'algebra', and 'calculus' merged into the broader label 'mathematics'. However in some cases,

the abstracts obtained both the broad and the detailed label. For instance, one abstract was labeled both 'humanities' and 'languages', despite 'languages' being a subset of the humanities discipline. This was done in order to keep consistency while still limiting the amount of labels to a maximum of five subjects per abstract.

After labeling all 300 abstracts manually, the importance of scientifically benchmarking the SLMs to GPT-4 was recognized. Hence, it was decided to further expand the dataset by adding the academic subjects classified by GPT-4. During this process, it became evident that GPT-4 successfully classified all abstracts with superior precision compared to our manual efforts. Consequently, it was ensured that if the SLMs identified the same subject as GPT-4 did, their answer would be considered correct, even if it varied from the initial manual categorization. For example, if a subject was manually categorized as "technology" and GPT-4 answered "information technology", both classifications were considered correct.

Summarization task

For the summarization task, the labels were chosen to be concise and correct summaries of the abstracts in the dataset, consisting of maximum 100 words. Due to both the impracticality of manually summarizing 300 abstracts within the time frame of this project and the possibility of human error, GPT-4 was used to perform this task. Moreover, the research was based on benchmarking to GPT-4, why this approach made sense. All of the summaries made by GPT-4 were however carefully reviewed and verified by us.

Translation task

The labels for the translation task are correct and full translations of the abstracts in the dataset from English to Swedish. GPT-4 was utilized to perform the translations, which were checked and corrected manually. This was done for 50 abstracts in the dataset.

3.2 The models

Five language models were selected for evaluation, each with a size around seven billion parameters, and all with a transformer architecture. All models were sourced from a website called HuggingFace.co, a company that develops computation tools for building applications using machine learning. The website serves as a platform where users share machine learning models and datasets with each other. During the last few years their library with transformer models has grown a lot, with over 350 000 models [58].

With the rapid pace of advancements in AI development in mind, only models launched within the past year were considered. Despite the cutting-edge nature of these models, it is important to note that the training processes usually takes several months, resulting in training data that is not always up to date. However, this is not something compromising the relevance of this thesis since the training data for subject classification, general text summarization and text translation from English to Swedish have been available during the training phases for all of the models in this evaluation.

The models chosen are all instruction-tuned, i.e. they are fine-tuned to be able to follow instructions well, similar to GPT-4. The fine-tuning is achieved by training the model on input-output pairs. The input is a description of an NLP task and the output is a demonstration of the wanted response. However, the instruction tuning does not teach the model any new knowledge other than how to follow instructions. Therefore, the model is always limited by the knowledge of its pre-trained version [59].

3.2.1 Mistral

Mistral, developed by MistralAI, is a generative transformer model trained on 7.24 billion parameters. The second instruction fine-tuned version, Mistral 7B v.2 Instruct, was used in this project. As reported by their developers in September 2023, the model surpasses the larger 13B model Llama 2, developed by Meta, across all evaluated benchmarks. Additionally, it outperforms the significantly larger Llama2 34B model in mathematics and coding [60]. The training data is not made public, but it is sourced from the open web. The MistralAI team justifies the choice of not publishing the dataset by referring to the competitive nature of the language model development field, opting to withhold the dataset from any rivals [61].

3.2.2 StableBeluga

The 6.74B version of StableBeluga2, derived from the Llama2 7B model, is heavily fine-tuned with open data. The model was released in July 2023 by the Stability AI team as a research project and does not yet have commercial licence. Its training methods were inspired from Microsoft's Ocra database, although different data sources were used [62]. Stability AI has reported that, in a range of benchmarks, StableBeluga2 demonstrates superior performance compared to GPT-3 [63] [64]. In the remainder of this thesis, StableBeluga2 will be referred to as StableBeluga.

3.2.3 OpenOcra

Alignment Lab AI, the creators of the dataset which StableBeluga is trained on, have also developed their own language model called Mistral 7B OpenOcra. As the name suggests, it is an enhancement of the Mistral 7B model, but fine-tuned using the OpenOcra dataset. In this thesis, the Mistral 7B OpenOcra model will be referenced to as OpenOcra in order to not confuse it with the Mistral model by MistralAI. OpenOcra was released in November 2023 [65].

3.2.4 Qwen

Version 1.5 of Qwen with 7.72 billion parameters, fine-tuned on a substantial dataset, was utilized. The specifics of the dataset are not publicly available. However, it is known that the model has been trained across 12 languages, including Arabic, Chinese, Japanese, Thai, and Russian. Released in February 2024, Qwen 1.5 stands as one of the latest additions to the field of text generation models. Its performance notably exceeds that of Llama2 7B in a variety of benchmarks, according to the Qwen team [66].

3.2.5 GPT-SW3

GPT-SW3 is an openly available transformer model created by AI Sweden, with the purpose to generate natural language. The model comes in several sizes, the largest being 40 billion parameters and the smallest 0.1 billion. The second version with a size of 6.7 billion parameters, fine-tuned for instructions, was used. GPT-SW3 differs itself from all other models by the fact that it is trained on both English and Swedish. The current version was released in May 2023 [9].

Table 3.1: Information about all evaluated models.

Model	**Size**	**Developer**	**Launch date**
GPT-SW3	6.7 B	AI Sweden	May 2023
Mistral	7.24 B	MistralAI	September 2023
OpenOcra	7 B	Alignment Lab AI	November 2023
StableBeluga	6.74 B	Stability AI	July 2023
Qwen	7.72 B	Qwen	February 2024

3.3 Prompt engineering

For all three different tasks, the instructions were written in the prompt. For the classification task, a list of all the possible subjects to classify the abstracts with was given in the prompt. Putting the list of subjects directly into the prompt was proven to be the most effective way to make the models understand which classes were allowed to choose from. The outlines of the prompts for the different tasks can be found in Appendix C.

The prompt engineering techniques discussed in the following sections were employed for the classification task only. Among the three tasks, the classification task emerged as the most challenging for the models to comprehend. This problem can likely be connected to the predefined list of classes, necessitating the assurance that the models recognized the limitation to only select from these predefined categories. In contrast, the tasks of summarization and translation appeared to be more straightforward. Hence, trying different prompting techniques was not prioritized for these tasks.

3.3.1 COSTAR prompting framework

In the formation of a suitable prompt for the classification task, the framework COSTAR, mentioned in Section 2.6.1 was initially employed. In Figure 3.1 the prompt following the COSTAR framework is provided.

Context
You are a professional classifier who is supposed to perform a classification task, choosing from a given list of classes.

Objective
The following is a document: "[abstract]". Based on this document, please identify the main themes. The ONLY themes you are allowed to choose from are: "[list of themes]". Please choose only a few themes from the list as the main themes of the text. ONLY choose themes from the list above. Do not come up with any other themes.

Audience
The main themes are supposed to be used for matching the abstracts with appropriate scholarships.

Response
Just provide the main themes as a response and nothing else.

Figure 3.1: Example prompt for classification task, following the COSTAR framework.

The "style" and "tone" were both eliminated directly. It was not considered necessary to specify these, since the desired response was supposed to consist only of the given themes and nothing else. For some models, the COSTAR framework resulted in hallucinated an-

swers or no answers at all. For other models it did not result in any apparent improvement in performance. Consequently, it was concluded that the COSTAR prompt was too long for the SLMs to comprehend.

3.3.2 Prompting SLMs using LLMs

When trying to format a prompt using the technique mentioned in Section 2.6.2, the prompt provided by GPT-4 was even longer than the one following the COSTAR framework. Consequently, this prompt was not used.

3.3.3 Prompting tags

For some of the models, specific prompting instructions were included in the models' research papers that are released when a new is launched. For example, it could be the usage of tags such as "system:", "user:", and "response:". In some cases, the models' performance improved once the prompts were modified according to the instructions in the research paper. Therefore, these tags were included in the prompts for some of the models.

3.3.4 Few-shot learning

Few-shot learning, as described in Section 2.6.3, was applied in the prompt for low-performing models. Five randomly chosen abstracts from the dataset and their corresponding labels were added into the prompt. However, for none of the models the added examples seemed to improve the performance noticeably. For this reason, this technique was not further proceeded with.

3.3.5 Chain-of-thought prompting

Given the difficulty to correctly classify the classes, it was explored whether the zero-shot classification ability of the models could be enhanced by incorporating CoT, as described in Section 2.6.4, reasoning in the prompt. The CoT prompt for zero-shot classification was built similar to the example shown in Figure 3.2.

The following is a document: "[abstract]". Based on this document, please identify the main themes. The ONLY themes you are allowed to choose from are: "[list of themes]".

Identify key features: Look for characteristics typically associated with each theme. If a theme, or a synonym to that theme, is mentioned in the document this could indicate that this should correspond to the main theme of the text.

Compare Against Known Patterns: Relate these features to what is commonly known about each theme. If a theme mentioned in the text often appears in the same context as another theme from the list this could also be considered as a main theme.

Weigh Evidence: Consider the presence or absence of features for each theme in the list and how they influence the classification.

Figure 3.2: Example prompt for zero-shot classification task, using CoT reasoning.

However, the CoT technique did not turn out to significantly improve the results. Therefore, a combination of one- and few-shot learning with CoT reasoning was applied. An example of such a prompt is shown in Figure 3.3.

The following is a document: "[abstract]". Based on this document, please identify the main themes. The ONLY themes you are allowed to choose from are: "[list of themes]".

For example, the "[text]" have the main themes "[mathematics]", "[technology]" and "[climate change]".

Identify key features: Since sustainability and the climate goals are mentioned in the text, [climate change] would be suitable as one main theme.

Compare Against Known Patterns: Since [technology] is often mentioned in the context of climate solutions this is also considered to be a main theme. Optimization is also mentioned in the text, which is a mathematical tool, hence [mathematics] is considered to be another main theme.

Weigh Evidence: Sentences involving climate change, technology and mathematics are all present in the given text. Hence, these are suitable as main themes.

Figure 3.3: Example prompt for one-shot classification task, using CoT reasoning.

The CoT did not improve the results for one- or few-shot classification either. In the end, the prompt that provided the best results was zero-shot, shorter and more concise. However, it is possible that the models became better at performing the task after a few rounds of prompting. The final prompt outlines used for the different tasks can all be found in Appendix C.

3.4 Coding details

All the code necessary for operating and assessing the models was developed in Python and can be found in Appendix B. To facilitate the loading of models and data, as well as to conduct evaluations using various metrics, several Python libraries were employed.

3.4.1 Python libraries

To load the data from PDF-files the *pypdf*-library was utilized. To download the models the *transformer*-library was used, using the packages *AutoModelForCausalLM*, *AutoTokenizer*, *AutoConfig*, *BitsAndBytesConfig* to load the models and build the pipeline. The pipeline includes model invocation, task specification, text input tokenization, and determination of the maximum number of tokens generated.

The evaluation process included using various packages from different machine learning libraries. From the *scikit*-library, the *f1_score* package was used to compute the corresponding metric. The *PyTorch*-library was used to calculate the perplexity metric, and the *Evaluate*-library was used to load the BLEU and the ROUGE score. These scores are further explained in Section 3.5.

3.4.2 Langchain framework

Langchain is a framework for development of applications powered by language models. Most importantly, it enables for applications to be context-aware. Context-aware applications connect a language model to sources of context such as prompt instructions or content to ground its response in. In the context of implementing and evaluating SLMs, Langchain was used to be able to connect the models to the dataset and prompt instructions.

The Langchain libraries used were *langchain* and *langchain-community*. Firstly, a simple LLMchain from the *langchain* library was used to connect the language model to the prompt. Secondly, a Text Loader from the *langchain-community* library was used to load in the data from a PDF file. Lastly, a *StuffDocumentsChain* from the *langchain* library connected the LLMchain to the extracted data from PDF-files.

3.5 Model evaluation

The performance of a machine learning model can be evaluated through a variety of metrics. For the purposes of this project, these metrics are classified into two primary categories; automatic evaluation and human evaluation. Automatic evaluations can be quantified and measured, providing objective metrics. On the other hand, human evaluations deliver a more subjective analysis, where individuals judge the model's outputs. Despite the quantifiable nature of automatic evaluation metrics, human evaluation holds equal, if not greater, significance. Given that the texts within the dataset are written by humans, the insight gained from human evaluators, who judge tasks they themselves could perform, proved to be very useful.

The models' multilingual capability was measured using both automatic and human evaluation. The focus was on testing the models' ability to comprehend Swedish prompts and to correctly translate texts from English to Swedish. Automatic evaluation metrics were then used both to measure how well the texts were translated from English to Swedish, compared to the labeled dataset, and to measure the fluency and coherence of the generated Swedish text. The prompt instructions, telling the models to translate the texts from English to Swedish, were given in Swedish. Thus, the capability of comprehending Swedish, generating a Swedish text, and translating from English to Swedish were all tested in the same task. Moreover, human evaluation was used to examine how well people, fluent in both English and Swedish, review the translations made by the models. For the GPT-SW3 model, the approach differed slightly. For this model, all of the prompts were written in Swedish and all of the text produced was in Swedish. This strategy was adopted because the model was specifically evaluated alongside the others to determine its proficiency in understanding and generating text in the Swedish language.

Additionally, one crucial factor for an overall understanding of the models was considered; their cost-effectiveness. For the purpose of this project, the cost-effectiveness involves the user cost for deploying the model in relation to the model's effectiveness, i.e. how well it performs and how fast it can be deployed. By combining these complementing metrics, a thorough overall evaluation of a model could be achieved. An overview of all of these evaluation metrics can be found Figure 3.4.

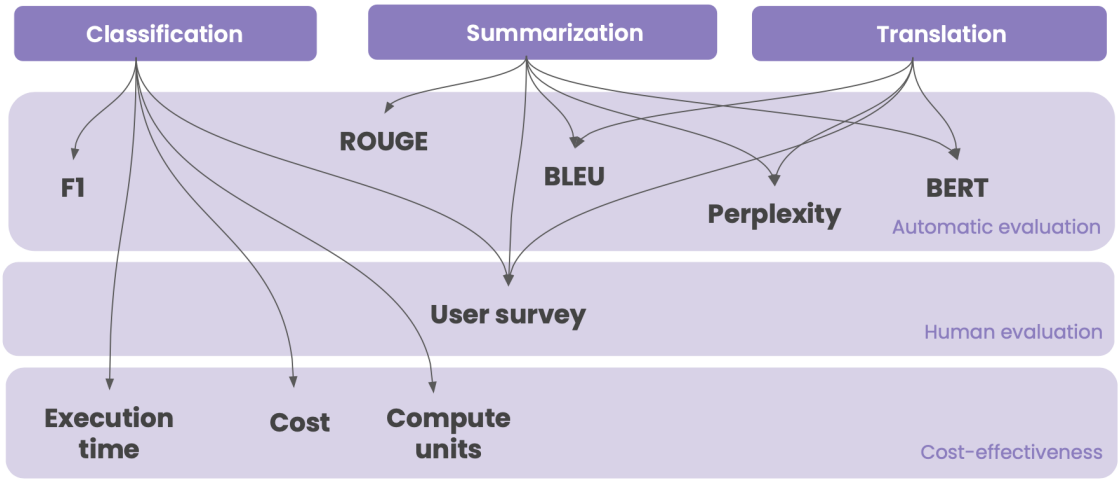


Figure 3.4: Overview of all evaluation metrics used for assessing the performance of the five SLMs.

3.5.1 Automatic evaluation

The automatic evaluation metrics used to assess the five models were the F1-score, ROUGE-1, ROUGE-2, BLEU, Perplexity, and BERT-F1. The AUC-ROC curve was excluded due to the difficulty of representing it with the large number of classes in this project. ROUGE-1, ROUGE-2, BLEU, and BERT-F1 were deemed sufficient for evaluating summarization and translation performance. METEOR, NIST, and LEPOR were excluded as they were not considered to provide significant additional information about the performance of the SLMs. Additionally, the WER score was deemed too general for the translation task and thus potentially misleading in the results.

F1-score

When evaluating multi-label classification using the F1-score, described in Section 2.7.1, the score was computed individually for each class, and the final F1-score was derived by averaging these scores.

ROUGE score

Only the ROUGE 1 and the ROUGE 2 scores, described in Section 2.7.1, were included among the ROUGE scores, since these two metrics combined are sufficient to cover the necessary aspects of the ROUGE scores.

However, relying solely on the ROUGE score is insufficient when evaluating a model's summarization skills, as there are always several valid summaries to a given abstract. For example, if an SLM wrote "This thesis describes" and a human or GPT-4 opts for "This thesis explains", the ROUGE score will not be able to account for such synonymy.

BLEU score

Although firstly developed to evaluate translations, the BLEU score, described in Section 2.7.1, also works well in assessing the performance of other NLP tasks, such as text summarization. Being easy to understand and implement, and not dependent on any specific language, the BLEU score is convenient to use when evaluating translations between English and Swedish. With this in mind, the BLEU score was used as an evaluation tool both for the translation and summarization tasks. In order to obtain the BLEU score for translation, the models' outputs can consist of only the translation. However, when performing the translation task, some of the models outputted additional text. Therefore, before calculating the BLEU score, the output had to be cleaned, keeping only the translation.

Perplexity

The perplexity score, introduced in Section 2.7.1, was calculated for both the translated Swedish texts and the English summaries.

BERT-F1 score

For the BERT score, described in Section 2.7.1, only the F1-score is presented in the result, since it captures both precision and recall.

3.5.2 Human evaluation

The aim of this project was ultimately to make the outputs from the models useful for humans, why it is important to measure how potential users would rate the performance of the models.

A user survey was carried out in order to receive feedback on the SLMs' capabilities regarding all three tasks; classification, summarization, and translation. The participants were 14 native Swedish potential users, fluent in English, who sent in their own theses' abstracts. They were then asked to judge the models' ability to catch the overlaying subject of their abstracts, as well as the models' capability to summarize and translate the abstracts. The subjects of the participants' thesis abstracts were from a variety of academic fields. For full details about the participants in the user survey, see Appendix

D. Key aspects considered were the relevance to the user, natural language flow in the English summaries and the Swedish translations, quality of the content, and the informative value. For all tasks, participants ranked each model's response on a scale from 1 to 4 or 5, with 1 being the best model and 4 or 5 the worst model. The lowest ranking for the classification and summarization task was 4, since GPT-SW3 was not included in these. However, in the translation task GPT-SW3 was included and therefore 5 was the lowest ranking. The reason for this was that GPT-SW3 produced Swedish classifications and summaries, which made the comparison with the other models' English outputs difficult.

The users were blinded to which model that corresponded to which response, ensuring unbiased judgement. It is important to note that, in the classification task, participants were not provided with the predetermined list of subjects. Consequently, they relied on their own judgment to assess the effectiveness with which the models categorized their abstracts. The output of GPT-4 was not included in the user survey. The user survey questions can be found Appendix D.

3.5.3 Cost-effectiveness

The cost-effectiveness of operating the SLMs is significantly influenced by factors such as runtime, processing units, and required memory. To monitor these aspects, the runtime, rate at which the units were consumed, and what run type used for each model were continuously recorded. In addition to estimating operation costs for all SLMs, the corresponding cost for performing the same tasks with GPT-4 was calculated, based on the implementation prices presented in Section 2.4.2. The prices used in the calculations were for the GPT-4 engine that Scholaro uses today, i.e. the standard model. This engine is $120 per 1 million sampled tokens, and the sampled tokens were estimated to be approximately 325 tokens per classifications. This number originates from an average value of the tokens sampled by the SLMs, where both the input abstract and the classifications were included.

Resource requirements

All the SLMs assessed were sourced from Hugging Face as open-source tools, free to access and utilize. Since the intended application for using these models is Scholaro's platform, which is meant to operate on laptops without any external computing and storage capacity, the decision was made to test the models in this manner. This approach was taken to determine the cost for Scholaro to use the SLMs without relying on external resources, since they do not have access to any.

An average laptop typically lacks the requisite GPU memory to operate locally, necessitating the use of cloud services. A Google Colab Pro subscription was chosen, priced at 11 euros per month and including 100 compute units. In addition to the Google Colab Pro subscription, a pay-as-you-go pricing model was adopted, based on the number of compute units necessary, where 100 units cost 11 euros. While running the models for the classification task, the amount of compute units used were noted. Compute units are what a GPU device consist of, and it usually has these units organized in a grid. The types of GPUs used to run the models were NVIDIA's V100 and A100 GPUs. The A100 GPU is the latest, most powerful GPU. However, the A100 was not always available as an option, therefore the V100 sometimes had to be used [67].

Execution time

It was further investigated how the execution time varied based on the selected run type, the construction of the prompt, and the code implementation strategies. The run type refers to the choice of processing unit to run the models on. As mentioned, both V100 and A100 GPUs were available options. Since the cost-effectiveness was supposed to be presented as cost per classification, the execution time was measured in seconds per classification. The reason for this was to be able to compare the cost with the time, aiming to get an accurate idea of the cost-effectiveness.

Chapter 4

Result

This chapter presents the results obtained from testing the five SLMs. It includes a table showing the automatic metrics, rankings from the user survey, and the cost, resources, and time required for each model. Additionally, comments from the user survey are provided.

4.1 Table

In Table 4.1, all the five models' scores are being presented. Marked in bold are the best achieved scores for each evaluation metric. All automatic scores are benchmarked to GPT-4, hence GPT-4 naturally scores perfect for the F1-score, ROUGE, BLEU, and BERT, i.e. 1.

Table 4.1: Model evaluation scores for all five evaluated models, sorted according to task. In bold are the best performing model according to each metric.

Evaluation scores for performance						
Model	Mistral	OpenOcra	StableBeluga	Qwen	GPT-SW3	GPT-4
CLASSIFICATION						
F1-score (0-1)	**0.81**	0.66	0.34	0.58	0.065*	1
User survey (1-4)	2.4	2.4	3.4	**1.1**	-	-
SUMMARIZATION						
ROUGE-1 (0-1)	**0.59**	0.58	0.50	0.48	0.44**	1
ROUGE-2 (0-1)	**0.29**	0.28	0.23	0.22	0.18**	1
BLEU-2 (0-1)	**0.39**	0.37	0.31	0.29	0.19**	1
BERT-F1 (0-1)	**0.92**	**0.92**	0.91	0.76	0.87**	1
User Survey (1-4)	2.2	**2.0**	2.4	3.2	-	-
TRANSLATION						
BLEU-2 (0-1)	0.47	0.50	0.02	0.35	**0.62**	1
BERT-F1 (0-1)	0.79	0.86	0.31	0.77	**0.90**	1
User Survey (1-5)	3.6	3.6	3.3	3.9	**1.7**	-
Perplexity SWE	7.7	8.4	**7.1**	13.1	18.92	-
Perplexity ENG	**13.2**	14.0	**13.2**	42.4	-	2.6

Cost-effectiveness and resource use						
Model	Mistral	OpenOcra	StableBeluga	Qwen	GPT-SW3	GPT-4
CLASSIFICATION						
Execution time (s/class.)	**34**	61	66	45	59	20
CU/class.	0.16	0.1	0.09	0.09	**0.07**	-
Cost (SEK/class.)	0.21	0.13	0.12	0.12	**0.09**	0.43

*The prompt for GPT-SW3 was written in Swedish, the list of allowed subjects was in Swedish, but the abstracts were in English.

**The prompt for GPT-SW3 was written in Swedish, and the summaries produced by the model were also in Swedish. For this task the abstracts were also in Swedish.

4.2 User survey comments

After ranking the models based on their performance in classification, summarization, and translation tasks, survey participants were offered the opportunity to comment on the models' outputs. A common critique concerning the classification task was that the categories were too broad. For example, one participant expressed a preference for the more specific category "Power Flow Optimization" as opposed to the broader "Energy Systems."

For the summarization task, a frequent comment was that many models did not produce genuine summaries but just paraphrased the original text. Qwen was particularly criticized for this, with remarks noting that it occasionally generated tokens in Chinese.

Regarding the translation task, two recurring criticisms emerged. Firstly, participants highlighted the models' difficulty in accurately translating complex academic vocabulary. For instance, one participant noted that no model successfully translated legal terminology, while another pointed out difficulties in translating specialized physics terms. Secondly, Qwen was specifically criticized for repeatedly producing tokens in Chinese, in this task as well.

Chapter 5

Discussion

This chapter begins by discussing the methodology of prompt engineering. It then addresses the choices for evaluating the models and the reliability for benchmarking their performance against GPT-4. Finally, it introduces areas for future research that would be of interest.

5.1 Prompting and usability

An interesting aspect of prompting techniques lies in the balance between crafting the perfect prompt and the time spent in doing so. In this thesis, it was decided that there existed a point in time when certain models had to be disregarded due to their lack of accuracy. Perhaps a different and very specific prompting technique would have enhanced the performance of these models. However, if an excessive amount of time is required to prompt correctly, the usability of the model may be questionable. Ultimately, a contributing factor to a model's effectiveness is its ability to comprehend various prompts, even if the prompts are not tailored to the model's specification.

One example of the critical importance of prompt formulation became evident in the decision regarding whether to number the list of subjects in a classification task or not. For certain models, numerical results were given as output, instead of subject names. These numbers were presumed to correspond to subjects from the list. Another example illustrating the need for specific prompting occurred with the use of tags from the model's Hugging Face card. These tags could be in the form of inserting the word "User:" or "System:" in the beginning of the prompt, or the word "Output:" at the end. If this kind of specific prompting is required, it could be argued, again, that the model is difficult to use.

5.2 Evaluation of the models

This section discusses the method of benchmarking the performance of the SLMs against GPT-4. Additionally, it covers the evaluation methods for all three different tasks and the evaluation metrics used.

5.2.1 Benchmarking against GPT-4

In the methodology of the evaluation, it was decided to benchmark against GPT-4. This decision was made because GPT-4 is currently used by Scholaro and is one of the best-performing LLMs for commercial use [68]. The benchmarking process involved labeling with GPT-4 during the creation of the dataset. Consequently, GPT-4 achieved perfect scores for all automatic evaluation metrics, except for perplexity, as shown in Table 4.1. This method was chosen for its consistency, as manual labeling efforts were considered insufficient.

One possibility could have been to include GPT-4's outputs in the user survey to validate the results with human evaluation. However, this approach was not taken, as it was considered more interesting to compare the results of the SLMs without the influence of outputs from an LLM.

Benchmarking against GPT-4 was considered valid for these experiments. However, it is important to recognize that SLMs may eventually match the performance of an LLM like GPT-4, making this benchmarking method less valid. At that point, manual labeling efforts may become more crucial. Nonetheless, considering the project's time-frame, manual labeling beyond the classification task was not reasonable for this study.

5.2.2 Classification

For the classification task a predetermined list of subjects to categorize abstracts was used. This was decided to be the most efficient method for deploying a classification tool aimed at matching scholarships with master's theses. Moreover, scholarship descriptions are usually not very specific. For instance, it is rare to encounter a scholarship explicitly for "Cardiology," whereas one broadly designated for "Medicine" is more common. This is why the models had to be limited to a list of broader subjects to classify from.

The classification task did not only test the models' ability to classify texts, but also their understanding of the prompt instructions, which specified that only certain classes were permissible. The latter aspect of the task was frequently overlooked by the models. It is

essential to recognize that a model may have accurately classified an abstract but failed to select amongst allowed subjects. Consequently, there is a variance between the results from the automatic scores and the user survey.

Participants in the survey were not provided with the predetermined list of subjects. This approach was chosen in order to not influence the participants and to get an input on the choice of labels. The findings from the user survey indicate that the labeling choices were too general. It appears that students favor more precise topics for their theses. One example supporting this theory is when one participant ranked the very narrow subdomains "Nonlinear Optics" and "Spectral broadening" higher than the broader subject "Physics". This preference may come from their familiarity with traditional keywords and a lack of understanding that scholarships frequently have broader themes in their descriptions. Consequently, it is evident that many students have an incorrect comprehension of scholarship mechanisms and the processes for obtaining them. Additionally, it is possible that the purpose of the classification task was not clear enough.

F1-score

As seen in Table 4.1, Mistral achieved the highest F1-score of 0.81. The benchmark was set by GPT-4, whose labels were used to compute the F1-scores for other models, thereby granting it a perfect F1-score of 1. OpenOcra and Qwen each registered scores below 0.7, StableBeluga fell below 0.4, and GPT-SW3 recorded a score below 0.1. Consequently, Mistral can be regarded as having performed somewhat satisfactorily in the classification task based on this metric.

A potential limitation in the methodology for calculating the F1-score may originate from the reliance on self-generated labels and those produced by GPT-4. It is possible that this labeling was not completely accurate, and alternative subjects might have been more appropriate. While employing an LLM for label generation is a recognized practice, it should not be solely relied upon without verification. The most significant limitation may be due to the fact that the list of subjects was created by us and GPT-4 was restricted to select from this list.

The approach of calculating the F1-score is adapted for scenarios where both outputs from the models and the correct labels are plural. As stated in Section 2.7.1, the method equates the model's performance in identifying at least one correct label amongst multiple predictions with that of a singular, accurate prediction. For instance, if the correct class is "law" and the model's output is "law, education, mathematics, physics", this is treated equivalently correct as only "law" as a prediction. This way of calculating the F1-score

was adopted as the most suitable solution to account for the model's range of different responses. Nonetheless, the downside of this method is that it favors a model's ability to case a wide net, potentially rewarding predictions including the correct label amongst several incorrect ones. In practice, it is preferable if the model pinpointed the correct label with great precision, rather than more or less guessing on an extensive number of subjects. A way to deal with this issue requires a solution that balances the need for precision with the reality that some abstracts genuinely hold multiple labels within similar academic fields.

The F1-score would be able to handle synonyms if these were categorized in the same class. However, in this project, only direct word matches were taken into account, meaning that models had to categorize texts under the exact subject provided as a benchmark to obtain a good score. For instance, the words "Sustainability" and "Sustainable Development" did not generate a match, even if it that would make sense in a scholarship matching process. Thus, in order to obtain a more thorough evaluation, it would be beneficial to have an additional F1-score with synonyms taken into account as well.

User survey results

In the user survey conducted to evaluate the classification task, as seen in Table 4.1, Qwen received the highest rating (1.1), followed by Mistral and OpenOcra (2.4). StableBeluga scored the lowest (3.4). Despite its lower F1-score, Qwen was considered the best model by the participants. Analysis of the survey responses revealed that the superior ratings were due to Qwen being both more specific and more numerous in its' classifications.

Qwen was not able to follow the prompting instructions of selecting an appropriate subject from the given list, but nevertheless, it was able to catch the overall theme of the abstract more accurately according to the user survey participants. This is an interesting observation since it points to the fact that there might have been a lack of subjects in the predefined list. Notably, the participants were not exposed to the predefined list of subjects, which could have influenced their ratings. This methodological choice was intended to determine whether the internal labeling and the curated list of subjects were sufficient. The results indicate that participants, who are most familiar with their theses, prefer a more specific subject description. It is possible that the outcome would have been different if the reviewers were not the ones who wrote the theses. However, it was considered interesting to see if the ones who could receive scholarships for their theses find the matching subjects suitable or not.

5.2.3 Summarization

For the summarization task, the benchmark summaries were exclusively generated by GPT-4 and subsequently verified by us. Therefore, all summarization scores were compared against these reference summaries. It is possible that the references were not flawless, which could be one source of error for all scores related to the summarization task.

Although the abstracts already are summaries of the master's theses, the summaries that are produced by the models are shorter and more concise ones, suitable for even shorter descriptions in Scholaro's database. Furthermore, the summarization task can be used for other purposes, such as compiling concise summaries of scholarship applications as described in Section 1.3.

ROUGE score

For the ROUGE-1 score, Mistral scored the highest with 0.59, followed by OpenOcra with 0.58. StableBeluga recorded a score of 0.50, while Qwen and GPT-SW3 both scored below 0.50. All models received scores above 0.4, indicating moderate performance for this metric. Particularly, the performance of Mistral, OpenOcra, and StableBeluga can be considered excellent. The high performance suggests that a significant number of words are identical between the summaries produced by the models and the reference summaries. However, this metric does not provide much insight into the flow of language, as it does not consider the context provided by surrounding words.

ROUGE-2 evaluates the match of bigrams to the reference summary, and it was expected that these scores would be lower than the ROUGE-1 scores due to the increased complexity of accurately reproducing bigrams, rather than unigrams. For the ROUGE-2 score, all models registered lower score as expected; Mistral led with 0.29, followed closely by OpenOcra with 0.28, while StableBeluga, Qwen, and GPT-SW3 each scored below 0.25. Thus, as stated in Section 2.7.1, these scores can not be considered good.

Both the ROUGE-1 and ROUGE-2 metrics evaluate only the exact overlap of unigrams or bigrams, respectively, and do not account for semantic or factual accuracy. Therefore, these metrics only serve as a useful preliminary measurement of how closely a summary aligns with the reference summary. Consequently, the summaries produced by all models could potentially be more accurate than the ROUGE scores implies.

BLEU-2 score

For the BLEU-2 score, Mistral achieved the highest with 0.39, followed by OpenOcra at 0.37, StableBeluga at 0.31, and both Qwen and GPT-SW3 scored below 0.30. Unlike ROUGE-2, BLEU-2 does not require the order of words in bigrams to be maintained, which accounts for the expected higher scores across all models compared to ROUGE-2. The BLEU-2 metric also primarily evaluates the overlap with reference summaries. However, it is more forgiving in its evaluation of grammatical structure, since it does not account for the order of words in the bigrams.

BERT-F1 score

In the evaluation of the BERT-F1 score, both Mistral and OpenOcra attained the highest rating of 0.92, followed by StableBeluga at 0.91, GPT-SW3 at 0.87, and Qwen at 0.72. The BERT-F1 metric accounts for synonyms, which explains the higher scores compared to the ROUGE and BLEU metrics across all models. However, in the analysis, the relevance of specific synonyms over others and the width of the synonym spectrum remains uncertain. Additionally, different grammatical structures can have the same semantic content. For instance, the sentences "Small language models are better than large language models" and "Small language models perform superior compared to large language models" have the same meaning, though the bigrams "perform superior" and "better than" would not be considered synonyms. This suggests that the summaries generated by the models might more closely resemble the reference summaries than the score indicates.

From these findings, it can be concluded that all models produced summaries that contain numerous synonyms relative to the reference summaries. However, whether the grammatical structures and the language flow in these summaries are sufficient cannot be directly determined from the BERT-F1 scores alone.

User survey results

The user survey indicates that OpenOcra generated the best summaries, followed by Mistral, StableBeluga, and Qwen. GPT-SW3's results were excluded from the user survey since its summaries were provided exclusively in Swedish, making them difficult for participants to compare to the English summaries. Notably, none of the models received a score higher than 2, with 1 being the highest possible rating. This suggests that participants did not regard any of the models' summaries as particularly high-quality. As mentioned in the results, participant feedback predominantly highlighted Qwen's poor performance and noted that some models did not generate real summaries, but instead merely paraphrased the original text.

5.2.4 Translation

For the translation task, the benchmark translations were generated by GPT-4 and proofread by us. All scores for this task are evaluated against these reference translations, similar to the summarization task. Consequently, it is possible that not all translations were flawless.

It is evident that the way the models are trained on different data affect their performance for this task. The greatest difference in training data amongst the five models is, while they are all primarily trained on English text, that GPT-SW3 is also trained on a vast amount of Swedish texts. This is most likely the reason why the GPT-SW3 model scores the highest in all of the following metrics.

BLEU-2 score

For the translation task, GPT-SW3 scored the highest at 0.62. This result can most probably be explained by the fact that GPT-SW3 is trained on an extensive amount of Swedish texts, and can thereby generate a more grammatically correct translation than the other models. Mistral and OpenOcra are not too far behind, with scores of 0.47 and 0.50 respectively, and Qwen scored the lowest with a value of 0.35. Since all scores over 0.3 are considered good, all these models produced translations somewhat acceptable.

StableBeluga scored an unexpectedly low score of 0.02, only successfully producing an output for about 30% of the total abstracts. Despite testing with multiple prompts, the specific cause for this result remains unclear. One theory is that the PDF Loader used to upload multiple abstracts simultaneously may not have been compatible with StableBeluga's requirement. Nevertheless, it is important to not solely focus on this low BLEU score, since StableBeluga's translations were more impressive according to the BERT score and the human evaluation.

BERT-F1 score

Given that the BERT-F1 score considers synonyms by measuring the cosine similarity across all Swedish vocabulary, it logically follows that the BERT F1-score exceeds the corresponding BLEU score for each model. The reason for this is, as mentioned, that BERT-F1 rewards the models when correctly choosing a synonym, and BLEU does not. Once again, GPT-SW3 scores the highest with a score of 0.90, a mark considered very good and nearly on par with GPT-4. OpenOcra, Mistral, and Qwen also perform well, with scores of 0.86, 0.79, and 0.77 respectively.

User survey results

GTP-SW3's performance in human evaluation aligns with the automatic results. According to the survey participants, GPT-SW3 scores the best with an average ranking of 1.7, where 1 represents the highest rating and 5 the lowest for this task. In contrast, all other models received scores ranging from 3.3 to 3.9 in the human evaluation. The consensus between both human and automatic evaluations underscores GPT-SW3 as the superior translator.

When assessing the results from the user survey, the models' struggle with the academic terminologies became evident. These difficulties were observed across academic fields such as biology, law, and journalism. The fact that this study was performed on academic texts, usually containing highly complex and specialized vocabulary, might have negatively influenced the model's translation scores. The models would likely be able to produce better translations if the original English texts were less complex. However, this would defeat the purpose of scholarship matching based on master's theses, but it is still interesting to note that they could be used for other practical applications.

Another notable aspect was that Qwen produced Chinese tokens at several occasions, something the survey participants commented on. This is due to the fact that Qwen is trained on several languages, Chinese included. It is possible that Qwen hallucinated during the translation task and defaulted to a language with which it was more familiar, rather than translating into Swedish.

5.2.5 Perplexity

The perplexity was evaluated for English summaries and Swedish translations. The findings indicated that the English perplexity scores were inferior when compared to the Swedish perplexity scores across mainly English-trained models, including Mistral, OpenOcra, StableBeluga and Qwen. For example, the perplexity score for Mistral was notably better in Swedish (7.7) versus its English counterpart (13.2). Keep in mind that a lower score is preferable, indicating a higher ability to predict the next word accurately.

At first sight, it seems odd that these four models appear to demonstrate higher coherence in Swedish than in English, considering they are primarily trained on English data. However, there exists an explanation based on the way perplexity is calculated. Perplexity relies on the word probability distribution function, $p(x)$, as described in Section 2.7.1. For instance, if a common word in English has a higher probability of appearing in a sequence, $p(x)$ will be higher. In contrast, the equivalent common word in Swedish

may have a lower $p(x)$ value, attributed to the limited fraction of Swedish data within the overall dataset. The perplexity score takes into account the sequence of two words, evaluating the likelihood of a subsequent word based on the prior word. This means that the models are less perplexed to see that the second word is another Swedish word if the first one is Swedish as well. In contrast, with a richer English vocabulary to draw from, predicting the next suitable English word becomes less probable and more challenging, thus increasing the perplexity score for English texts.

Similarly, the perplexity score for the Swedish-trained model, GPT-SW3, can be explained. Although it is evident that GPT-SW3 is superior in generating Swedish test according to the human evaluation, BLEU, and BERT-F1, its perplexity score is higher (indicating poorer performance) compared to the English-trained models. Since GPT-SW3 is trained on larger amounts of Swedish and Nordic texts than the other models, GPT-SW3 is more perplexed by the encounter of any arbitrary Swedish word followed by another. The Swedish vocabulary is broader for GPT-SW3, and there exists a greater amount of suitable words to follow up another Swedish word. Essentially, GPT-SW3 is more "surprised" by the sequence of Swedish words it generates, hence the higher perplexity score, even though it outperforms its English-trained counterparts in terms of generating coherent Swedish text.

In conclusion, it is important to keep in mind that the perplexity score is significantly influenced by the type and volume of data on which the models are trained, particularly in terms of English and Swedish content. Therefore, the Swedish perplexity scores should be used for comparing the models against each other, rather than for comparing a single model's English and Swedish proficiency. Additionally, the Swedish perplexity score for GPT-SW3 should not be directly compared to those of English-trained models or to its own English perplexity score. Instead, it could serve as a benchmark for evaluating the performance of other Swedish-trained SLMs in the future.

5.2.6 Cost-effectiveness and resource use

In Section 3.5.3 the time, computational units, and cost per classification were recorded for each model. Mistral had the lowest execution time, averaging 34 seconds per classification. This compares to GPT-4, which required 20 seconds to classify one thesis abstract. The second quickest was Qwen, averaging 45 seconds per classification. OpenOcra, StableBeluga, and GPT-SW3 each took approximately one minute per classification. These durations are significant when evaluating the models' efficiency. Naturally, a quicker model is preferable. However, it is essential to consider execution time in combination with the model's performance efficacy.

It is possible that factors other than the inherent speed of the models could have influenced execution times. For instance, the design of the prompts might have impacted time efficiency, which could have been a reason to why longer execution times occasionally were observed with more complex prompts. Nonetheless, as the same prompts were given to all models for identical tasks, the comparison among the models should remain unaffected by the prompt construction. Similarly, the implementation of the models used the code framework based on Langchain to construct sequences, which was identified as one of the most efficient methodologies available. However, alternative approaches might exist that could potentially reduce execution times further.

Resource usage was measured in compute units per classification. Notably, Mistral consumed the most resources, while GPT-SW3 used the least. Nevertheless, when correlated with their respective F1-scores, it is evident that Mistral significantly outperforms GPT-SW3 in terms of performance in this task. Given that compute units are directly associated with the cost per classification, Mistral emerges as the most expensive model, whereas GPT-SW3 is the cheapest. However, Mistral is priced at 0.21 SEK per classification, which is approximately half the cost of GPT-4, priced at 0.43 SEK per classification.

When operating the models within the Google Colab environment, the cost estimations were based on the fact that 100 GPU units are priced at 11 euros (130 SEK), thus aligning the cost directly with GPU usage for each run. However, as detailed in Section 3.5.3, costs can vary significantly depending on what type of environment, local servers or virtual computers that are accessible. If one were to use powerful local servers or virtual computers, the costs would differ significantly and would not be directly proportional to compute unit consumption. Therefore, selecting the most appropriate environment that aligns with one's resources and purposes is important to maximize cost-effectiveness.

5.3 Future research

Certain important evaluation aspects could not be encompassed within the scope of this master's thesis, such as energy consumption, data security, data governance, and bias. Therefore, these metrics are recommended for future research. Additionally, further evaluation metrics that could provide more comprehensive information on model performance, either untested or yet to be developed, are under consideration. Furthermore, the potential to fine-tune base models with custom datasets and to combine several SLMs is discussed.

5.3.1 Energy consumption

Firstly, it would be interesting to research how much energy was consumed, and thereby how many carbon equivalents were released, in the training phase of each model. This would be valuable information to be able to make environmental sustainable model choices.

It is safe to state that the models' most significant environmental impact origins from the training phase, due to their reliance on extensive datasets over long periods. The energy consumption required for training far outweighs the consumption associated with testing these models on our data. However, obtaining precise information on the energy consumption required for training the different models proved to be difficult. For certain models, details on the carbon footprint associated with their training were provided on Hugging Face. In other cases, this information was disclosed in the accompanying research papers. Notably, larger organizations, such as Google and Meta, tend to report this data more frequently, possibly due to their sustainability initiatives. Nevertheless, none of the models that were evaluated had any accessible information concerning the energy consumption or carbon footprint associated with their training phase.

Since information on energy consumption during training was difficult to find, the intention was initially to estimate the energy consumption of each model during its testing phase instead. However, it was recognized that additional variables, beyond the scope of this study, significantly affect energy consumption estimates. Importantly, the type of compute units used can substantially influence energy usage. The approach to use V100 and A100 GPUs was not systematic, as the choice depended on their availability. Additionally, information regarding the energy sources that powered these compute units could not be found. Nonetheless, it is reasonable to assume that more compute units emit more CO2.

5.3.2 Data privacy

Secondly, another interesting aspect to keep in mind when evaluating language models is their data security and data privacy. This is a scope that lies beyond this master thesis, since no evaluation metric that could capture different models' capability to securely store data input could be found. If such a metric would exist, it would be interesting to look into aspects such as if, and for how long, the models save input data. Additionally, where the servers are located, and how the security systems around the models work would be of interest. In general, it can be argued that smaller models, with the ability to run locally, are more secure than larger models in need of being operated on cloud services. This gives an advantage of SLMs over LLMs in the aspect of data privacy.

5.3.3 Data governance

Thirdly, an additional evaluation metric to consider when selecting models is data governance, which includes the nature of language produced by the models and their potential of generating toxic outputs. An example could be the models' capability to produce hazardous content, such as providing instructions on how to carry out illegal actions. Models will never be better or more secure than the data they are trained on. Thus, since the training data can contain any type of content, the models go under a process called red-teaming. This process includes teaching the models to avoid hazardous content and is based on letting humans read prompts and classify them as toxic or not [69]. It would be interesting to explore further how well these SLMs are governed compared to LLMs, by using a quantified evaluation metric.

5.3.4 Bias

Lastly, an interesting aspect to consider when choosing between language models is their potential level of bias. A biased model refers to outputs that do not accurately reflect reality, but instead skew towards one direction. An AI model can become biased at any of the training stages, but bias usually originates during the data collection phase. Essentially, if the training data is incomplete or unrepresentative, it may wrongly favor certain outcomes over others [70]. For example, in the health care sector, it has been reported that datasets have underrepresented women and minority groups. This can lead to lower accuracy rates of computer-aided-diagnosis systems for these groups [71]. In this study, it is possible that models favoured certain classifications over others, or that the translations were biased, reflecting the skewed vocabulary present in the training data. However, it is not evident how the models could be biased toward a certain group of people in the

tasks evaluated in this thesis. Potentially, there could be a bias issue with other aspects within the scholarship matching process, such as favoring a certain group of scholarship candidates. Thus, guaranteeing unbiased models are more important for other tasks in the scholarship matching process that lies outside the scoop of this paper.

5.3.5 Fine tuning with custom dataset

An avenue for future research involves further development of SLMs by fine-tuning them on pre-existing base models. This would improve the SLM's capabilities in task-specific operations to be able to classify, summarize, and translate with higher precision. However, the fine-tuning would require an extensive dataset containing abstracts labeled with their corresponding subjects, reference summaries, and reference translations. The dataset of 300 subjects and 50 reference summaries and translations is not near large enough. However, it would potentially be possible to gather a greater dataset over a longer period of time, either manually or by letting users signing up on the Scholaro site submit their abstract along with their corresponding subjects. If a dataset large enough and with acceptable quality was to be obtained, non-instruct base models such as Gemma or Llama 2 could be fine-tuned to obtain a model specified for abstract classification, summarization or translation.

5.3.6 Combination of several SLMs

Another area that warrants further investigation is the potential for combining multiple SLMs. As discussed in previous sections, some of the SLMs evaluated in this thesis excelled in different tasks. Therefore, to maximize the utility of an application, one could possibly employ one model for classification, one for summarization, and another for translation. In such a scenario, different prompts could be tailored for each model. However, due to time constraints, this project did not investigate the optimal way to combine the models. Exploring this aspect in future research would be interesting and potentially beneficial to implement in an application such as Scholaro's platform.

Chapter 6

Conclusion

Going through the research questions one by one, the following were the conclusions.

Which automatic and human evaluation metrics are most effective for assessing the performance of an SLM's ability to classify, summarize and translate texts from English to Swedish?

Both automatic and human evaluation metrics proved to be suitable for determining the performance of SLMs. Furthermore, it was concluded that a variety of evaluations metrics were necessary for accurately performing a full evaluation, since the models ranked differently for different metrics. The automatic metrics chosen were:

- For classification: F1-score.
- For summarization: ROUGE-1 score, BLUE-2 score, BERT-F1 score, and perplexity.
- For translation: BLUE-2 score, BERT-F1 score, and perplexity.

For the human evaluation, the most effective metric was chosen to be a user survey, in order to get human feedback and to complete the automatic evaluation metrics.

In terms of these evaluation metrics, how do SLMs perform compared to each other and to GPT-4 when classifying, summarizing, and translating master's theses?

After testing the SLMs on around 300 abstracts, including the ones used for the user survey, the findings indicated that the effectiveness of the SLMs evaluated varies

significantly across the different tasks classification, summarization, and translation. Furthermore, it was concluded that if the right SLM is chosen for a suitable task, together with the correct prompting techniques, some of the models can indeed compete with their larger counterpart GPT-4.

Which evaluation metrics could be used to assess the SLMs' bilingual capability to comprehend Swedish prompts and generate Swedish text, and how well do the SLMs perform in this area compared to each other and to GPT-4?

Overall, the models demonstrated a rather limited ability to comprehend and generate Swedish texts. The outstanding model was GPT-SW3, which received the highest scores for all automatic evaluation metrics in the translation task, except for perplexity. It also received an outstanding ranking in the user survey.

Apart from the automatic and human evaluation metrics, what other surrounding factors could be considered when choosing SLMs for these tasks? How can they be measured and how do the SLMs compare to each other and to GPT-4?

The main surrounding factor that could be considered during the time-frame of this project was the cost-effectiveness. The time, resource requirements, and cost of deploying the models were measured by counting compute units per classification. All of the SLMs were much cheaper to deploy compared to GPT-4, with the most expensive SLM being less than half the price of GPT-4. However, GPT-4 was concluded to be faster than all the evaluated SLMs.

In conclusion, the evaluation of SLMs across the chosen tasks demonstrates the necessity of employing both automatic and human metrics. The selected metrics provided a broad understanding of strengths and weaknesses specific to each model in the tasks classification, summarization, and translation. Particularly, the results emphasize the capabilities of the models Mistral, OpenOcra, and GPT-SW3 to compete with GPT-4 in certain tasks. Mainly, Mistral and OpenOcra were on par with GPT-4 for the classification and summarization task, and GPT-SW3 for the translation task. The human evaluation provided essential insights into the models' effectiveness, particularly in generating Swedish text. This approach of combining different metrics concludes the importance of thorough evaluation to understand the potential of SLMs in practical applications, such as scholarship matching.

References

[1] Viktorija Pešić and Sofia Lindeberg. Stockholms studentbudget 2024, January 2024. (SSCO) https://www.ssco.se/rapporter-1/stockholms-student-budget-2024 Visited on 5/6/2024.

[2] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *Findings of ACL 2023*, July 2023. https://doi.org/10.48550/arXiv.2305.02301.

[3] Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving $>$ 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*, December 2023. https://doi.org/10.48550/arXiv.2312.09241.

[4] Timo Schick and Hinrich Schütze. It's not just size that matters small language models are also few-shot learners. *NAACL 2021*, April 2021. https://doi.org/10.48550/arXiv.2009.07118.

[5] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*, November 2023. https://doi.org/10.48550/arXiv.2311.11045.

[6] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, August 2023. https://doi.org/10.48550/arXiv.2305.17926.

[7] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Large language models are not fair evaluators, 2023. (HuggingFace) https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard Visited on 5/6/2024.

[8] Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*, June 2023. https://doi.org/10.48550/arXiv.2306.04757.

[9] Ariel Ekgren, Amaru Cuba Gyllensten, Felix Stollenwerk, Joey Öhman, Tim Isbister, Evangelia Gogoulou, Fredrik Carlsson, Alice Heiman, Judit Casademont, and Magnus Sahlgren. Gpt-sw3: An autoregressive language model for the nordic languages. *arXiv preprint arXiv:2305.12987*, May 2023. https://doi.org/10.48550/arXiv.2305.12987.

[10] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *Machine Learning, A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[11] Mattias Ohlsson and Patrik Edén. *Introduction to Artificial Neural Networks and Deep Learning*. Lund University, 2022.

[12] F. Bre, J. M. Giménez, and V. Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildnings*, 158:1429–1441, 2018.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems 30*, June 2017. https://doi.org/10.48550/arXiv.1706.03762.

[14] Chenguang Wang, Mu Li, and Alexander J. Smola. Language models with transformers. *arXiv preprint arXiv:1904.09408*, April 2019. https://doi.org/10.48550/arXiv.1904.09408.

[15] Luis Serrano. What are transformer models and how do they work?, 2023. (Cohere) https://cohere.com/blog/what-are-transformer-models Visited on 5/6/2024.

[16] Eram Munawwar. A comprehensive guide to subword tokenisers, 2020. (Towards Data Science) https://towardsdatascience.com/

a-comprehensive-guide-to-subword-tokenisers-4bbd3bad9a7c Visited on 5/6/2024.

[17] S. Joshua Johnson, M. Ramakrishna Murty, and I. Navakanth. A detailed review on word embedding techniques with emphasis on word2vec. *Multimedia Tools and Applications*, October 2023. https://doi.org/10.1007/s11042-023-17007-z.

[18] Richmond Alake. Understanding cosine similarity and its applications, 2023. (Built in) https://builtin.com/machine-learning/cosine-similarity Visited on 5/6/2024.

[19] Deepanshusachdeva. Understanding transformers step by step — word embeddings, 2023. (Medium) https://deepanshusachdeva5.medium.com/understanding-transformers-step-by-step-word-embeddings-4f4101e7c2f Visited on 5/6/2024.

[20] Enes Zvornicanin. What are embedding layers in neural networks?, 2024. (Baeldung) https://www.baeldung.com/cs/neural-nets-embedding-layers Visited on 5/6/2024.

[21] MLNerds. What is the difference between word2vec and glove ?, 2019. (Machine Learning Interview) https://machinelearninginterview.com/topics/natural-language-processing/what-is-the-difference-between-word2vec-and-glove/ Visited on 5/6/2024.

[22] Maxime. What is a transformer?, 2019. (Medium) https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04 Visited on 5/6/2024.

[23] Wikipedia. Softmax function, 2024. (Wikipedia) https://en.wikipedia.org/wiki/Softmax_function Visited on 5/6/2024.

[24] Cloudflare. What is a large language model (llm)?, 2023. (Cloudflare) https://www.cloudflare.com/learning/ai/what-is-large-language-model/# Visited on 5/6/2024.

[25] Asif Razzaq and Arham Islam. Top large language models (llms) in 2023 from openai, google ai, deepmind, anthropic, baidu, huawei, meta ai, ai21 labs, lg ai research and nvidia, 2023. (Mark Tech Post) https://shorturl.at/OLaAt 5/6/2024.

[26] Omega Venture Partners. Introduction to large language models, 2022. (Omegavp) https://www.omegavp.com/articles/introduction-to-large-language-models/ Visited on 5/6/2024.

[27] José Antonio Ribeiro Neto. Chatgtp and the generative ai hallucinations, 2023. (Medium) https://medium.com/chatgpt-learning/chatgtp-and-the-generative-ai-hallucinations-62feddc72369 Visited on 5/6/2024.

[28] OpenAI. How much does gpt-4 cost?, 2024. (Open AI) https://help.openai.com/en/articles/7127956-how-much-does-gpt-4-cost Visited on 5/6/2024.

[29] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15, November 2022. https://doi.org/10.48550/arXiv.2211.02001.

[30] Alexandra Sasha Luccioni and Alex Hernandez-Garcia. Counting carbon: A survey of factors influencing the emissions of machine learning. *arXiv preprint arXiv:2302.08476*, February 2023. https://doi.org/10.48550/arXiv.2302.08476.

[31] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, and William Bergeron. From words to watts: Benchmarking the energy costs of large language model inference. *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, September 2023.

[32] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, April 2021. https://doi.org/10.48550/arXiv.2104.10350.

[33] Nagesh Mashette. Small language models (slms), 2023. (Medium) https://medium.com/@nageshmashette32/small-language-models-slms-305597c9edf2 Visited on 5/6/2024.

[34] Tanya Malhotra. Everything you need to know about small language models (slm) and its applications, 2023. (Mark Tech Post) https://shorturl.at/dwUZn Visited on 5/6/2024.

[35] Bijit Ghosh. The rise of small language models— efficient customizable, 2023. (Medium) https://medium.com/@bijit211987/the-rise-of-small-language-models-efficient-customizable-cb48ddee2aad Visited on 5/6/2024.

[36] Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4. *arXiv preprint*

arXiv:2312.16171, December 2023. https://doi.org/10.48550/arXiv.2312.16171.

[37] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*, June 2023. https://doi.org/10.48550/arXiv.2306.04528.

[38] Frugal Zentennial. Unlocking the power of costar prompt engineering: A guide and example on converting goals into system of actionable items, 2024. (Medium) https://shorturl.at/voalF Visited on 5/6/2024.

[39] Jordan Gibbs. Forget prompt engineering, chatgpt can write perfect prompts for you, 2024. (Medium) https://shorturl.at/RoQSb Visited on 5/6/2024.

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, January 2023. https://doi.org/10.48550/arXiv.2201.11903.

[41] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(39):1–45, March 2024. https://doi.org/10.1145/3641289.

[42] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024. https://arxiv.org/pdf/2307.03109.

[43] Leon Derczynski. Complementarity, F-score, and NLP evaluation, May 2016. (Aclanthology) https://aclanthology.org/L16-1040.pdf Visited on 5/6/2024.

[44] Eren Kızılırmak. Text summarization: How to calculate rouge score, 2023. (Medium) https://medium.com/@eren9677/text-summarization-387836c9e178 Visited on 5/6/2024.

[45] Sthanikam Santhosh. Understanding BLEU and ROUGE score for NLP evaluation, 2023. (Medium) https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb Visited on 5/6/2024.

[46] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, July 2002. https://dl.acm.org/doi/epdf/10.3115/1073083.1073135.

[47] Chiara Campagnola. Perplexity in language models, 2020. (Towards Data Science) https://towardsdatascience.com/perplexity-in-language-models-87a196019a94 Visited on 5/6/2024.

[48] Lukas Görög. Exploring the latest advancements in gpt-4: A comprehensive overview, 2023. (Predictea) https://www.predictea.com/exploring-the-latest-advancements-in-gpt-4-a-comprehensive-overview/ Visited on 5/6/2024.

[49] Priyanka. Perplexity in language models, 2022. (Medium) https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72 Visited on 5/6/2024.

[50] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, February 2020. https://doi.org/10.48550/arXiv.1904.09675.

[51] İlyurek Kılıç. Roc curve and auc: Evaluating model performance, 2023. (Medium) https://medium.com/@ilyurek/roc-curve-and-auc-evaluating-model-performance-c2178008b02 Visited on 17/6/2024.

[52] Lavie Alon and Michael J. Denkowski. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23:105–115, November 2009. https://doi.org/10.1007/s10590-009-9059-4.

[53] Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting bleu/nist scores: How much improvement do we need to have a better system? *LREC*, 2004. http://www.lrec-conf.org/proceedings/lrec2004/pdf/755.pdf.

[54] Aaron L.F. Han, Derek F. Wong, and Lidia S. Chao. Lepor: A robust evaluation metric for machine translation with augmented factors. *Proceedings of COLING*

2012: Posters, pages 441–450, 2012. https://aclanthology.org/C12-2044.pdf.

[55] Stephen M. Walker II. What is the word error rate (wer) score? (KLU) https://klu.ai/glossary/wer-score Visited on 17/6/2024.

[56] Digitala vetenskapliga arkivet. Digitala vetenskapliga arkivet, 2024. (Digitala vetenskapliga arkivet) https://www.diva-portal.org/smash/search.jsf?dswid=9239 Visited on 5/6/2024.

[57] Digitala vetenskapliga arkivet. Open access, 2023. (Digitala vetenskapliga arkivet) https://www.info.diva-portal.org/om-diva/open-access/ Visited on 5/6/2024.

[58] HuggingFace. Text summarization: How to calculate rouge score, 2024. (HuggingFace) https://huggingface.co/docs/hub/index Visited on 5/6/2024.

[59] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, March 2022. https://doi.org/10.48550/arXiv.2203.02155.

[60] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, October 2023. https://doi.org/10.48550/arXiv.2310.06825.

[61] Arthur Mensch. Training data?, 2023. (HuggingFace) https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/8 Visited on 5/6/2024.

[62] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, June 2023. https://doi.org/10.48550/arXiv.2306.02707.

[63] Dakota Mahan, Ryan Carlow, Louis Castricato, Nathan Cooper, and Christian Laforte. Stable beluga models, 2023. (HuggingFace) https://huggingface.co/stabilityai/StableBeluga2 Visited on 5/6/2024.

[64] Stability.ai. Meet stable beluga 1 and stable beluga 2, our large and mighty instruction fine-tuned language models, 2023. (Stability AI) https://stability.ai/news/stable-beluga-large-instruction-fine-tuned-models Visited on 5/6/2024.

[65] Wing Lian, Bleys Goodson, Guan Wang, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Mistral-7b-openorca, 2023. (HuggingFace) https://huggingface.co/Open-Orca/Mistral-7B-OpenOrca Visited on 5/6/2024.

[66] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Fei Huang, Binyuan Hui, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Tianyu Liu, Keming Lu, Jianxin Ma, Rui Men, Na Ni, Xingzhang Ren, Xuancheng Ren, Zhou San, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Jin Xu, An Yang, Jian Yang, Kexin Yang, Shusheng Yang, Yang Yao, Bowen Yu, Jianwei Zhang, Yichang Zhang, Zhenru Zhang, Bo Zheng, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Introducing qwen1.5, 2024. (GitHub) https://qwenlm.github.io/blog/qwen1.5/ Visited on 5/6/2024.

[67] NVIDIA. Nvidia cloud and data center, 2024. (NVIDIA) https://www.nvidia.com/en-us/data-center/a100/ Visited on 5/6/2024.

[68] OpenAI. Gpt-4 technical report. *ArXiv*, 2024. https://arxiv.org/pdf/2303.08774.

[69] Adam Zewe. A faster, better way to prevent an ai chatbot from giving toxic responses, April 2024. (MIT News) https://news.mit.edu/2024/faster-better-way-preventing-ai-chatbot-toxic-responses-0410 Visited on 5/6/2024.

[70] Chapman-University. Bias in AI. (Chapman University) https://www.chapman.edu/ai/bias-in-ai.aspx#:~:text=Types%20of%20Bias%20in%20AI&text=Selection%20bias%3A%20This%20happens%20when,lead%20to%20an%20unrepresentative%20dataset. Visited on 5/6/2024.

[71] IBM. Shedding light on ai bias with real world examples, 2023. (IBM) https://www.ibm.com/blog/shedding-light-on-ai-bias-with-real-world-examples/ Visited on 5/6/2024.

Appendices

Appendix A

The labeled dataset

The dataset is uploaded to Hugging Face and is openly available under the Apache 2.0 licence.

Link to the labeled dataset: Hugging Face Dataset
(URL: https://huggingface.co/datasets/adminscholaro/abstracts)

List of allowed subjects

1. Archaeology
2. Architecture
3. Artificial intelligence
4. Biology
5. Business
6. Chemistry
7. Civil engineering
8. Climate change
9. Communication
10. Computer science
11. Cultural studies
12. Design
13. Economics
14. Education
15. Electrical engineering
16. Energy systems
17. Entrepreneurship
18. Fashion
19. Finance
20. Gender equality

21. Gender studies
22. Geography
23. Global health
24. Healthcare
25. Healthcare systems
26. History
27. Human resources
28. Human rights
29. Humanities
30. Industrial management
31. Information technology
32. Infrastructure
33. Integration
34. Language
35. Law
36. Literature
37. Machine learning
38. Marketing
39. Mathematics
40. Mechanical engineering
41. Media studies
42. Medicine
43. Military science
44. Music
45. Organizational management
46. Pharmaceuticals
47. Philosophy
48. Physics
49. Political science
50. Project management
51. Psychology
52. Religion
53. Social science
54. Social structures
55. Social work
56. Sports
57. Sustainability
58. Technology
59. Urban planning

Appendix B

The scripts in Python

Classification script outline:

```
# Installing packages
pip install --upgrade --quiet  langchain-openai tiktoken chromadb
langchain accelerate transformers
pip install pypdf

import transformers
import torch
import numpy as np
import re

from transformers import AutoModelForCausalLM, AutoTokenizer,
    AutoConfig, BitsAndBytesConfig
from langchain import PromptTemplate, LLMChain
from langchain.llms import HuggingFacePipeline
from langchain.chains.summarize import load_summarize_chain
from langchain.chains.combine_documents.stuff import
    StuffDocumentsChain
from langchain_community.document_loaders import PyPDFLoader,
    TextLoader
from langchain.text_splitter import CharacterTextSplitter
from sklearn.metrics import f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import precision_score, recall_score, f1_score

# Data
subjects = [LIST OF SUBJECTS] # Predefined subjects from list
```

```
data = [LABELS] # Correct labels to each abstracts
correct = data.strip().split("\n")

# Functions
def remove_newlines(vector):
    return [string.replace('\n', ', ') for string in vector]

def filter_answers(answers, subjects):
    filtered = []
    lower_subjects = [subject.lower() for subject in subjects]
    for answer in answers:
        words = re.findall(r'\b[\w\s]+\b', answer)
        common_words = ', '.join(word.strip() for word in words if word.strip().lower() in lower_subjects)
        if common_words:
            filtered.append(common_words.lower())
        else:
            filtered.append(" ")
    return filtered

def calculate_f1_score(result_vector, correct_vector):
    true_positives = 0
    false_positives = 0
    false_negatives = 0

    for result, correct in zip(result_vector, correct_vector):
        result_subjects = set(result.split(", "))
        correct_subjects = set(correct.split(", "))
        result_subjects.discard('')
        correct_subjects.discard('')
        matches = result_subjects.intersection(correct_subjects)
        if matches:
            true_positives += 1  # Correct match found
        else:
            if result_subjects:  # Result predicted something but it was wrong
                false_positives += 1
            if correct_subjects:  # Correct had something but result predicted nothing or wrong
                false_negatives += 1
    length_difference = abs(len(result_vector) - len(correct_vector))
    false_negatives += length_difference

    precision = true_positives / (true_positives + false_positives) if (true_positives + false_positives) > 0 else 0
```

```
    recall = true_positives / (true_positives + false_negatives) if (true_positives + false_negatives) > 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    return precision, recall, f1

# Loading the model
model = AutoModelForCausalLM.from_pretrained('HUGGING FACE MODEL ID', trust_remote_code=True)
tokenizer = AutoTokenizer.from_pretrained('HUGGING FACE MODEL ID')
pipeline = transformers.pipeline(
    model=model,
    tokenizer=tokenizer,
    task="text-generation",
    eos_token_id=tokenizer.eos_token_id,
    pad_token_id=tokenizer.eos_token_id,
    repetition_penalty=1.1,
    return_full_text=True,
    max_new_tokens=25,
)

# Prompting
prompt_template = "PROMPT HERE"
prompt = PromptTemplate.from_template(prompt_template)
llm = HuggingFacePipeline(pipeline=pipeline)
llm_chain = LLMChain(llm=llm, prompt=prompt)

# Loading the abstracts
loader = PyPDFLoader("ABSTRACT PDF URL")
abstract_load = loader.load()
stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
   document_variable_name = "abstract")
answers = []

# Run the model on the abstract
for i in range(47):
  abstract = [(abstract_load[i])]
  stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
   document_variable_name = "abstract")
  theme = stuff_chain.run(abstract)
  answers.append(theme)
  print('Theme',i+1,theme)

answers = remove_newlines(answers) # Remove newlines, preprocessing
result = filter_answers(answers, subjects) # Remove all words that
```

```
    are not subjects

# Calculating F1 score
[precision, recall, f1] = calculate_f1_score(correct, result)
print(f'F1 score: {f1}')
```

Summarization script outline:

```
# Installing packages
pip install --upgrade --quiet  langchain-openai tiktoken chromadb
    langchain accelerate transformers
pip install evaluate
pip install pypdf

import transformers
import torch
import numpy as np
import evaluate

from transformers import AutoModelForCausalLM, AutoTokenizer,
    AutoConfig, BitsAndBytesConfig
from langchain import PromptTemplate, LLMChain
from langchain.llms import HuggingFacePipeline
from langchain.chains.summarize import load_summarize_chain
from langchain.chains.combine_documents.stuff import
    StuffDocumentsChain
from langchain_community.document_loaders import PyPDFLoader,
    TextLoader
from langchain.text_splitter import CharacterTextSplitter
from evaluate import load

import os
os.environ['HF_TOKEN'] = 'HUGGING FACE PERSONAL TOKEN'

# Data
correct_summaries = [LABELED SUMMARIES]

# Prompting
prompt_template = "YOUR PROMPT HERE"
prompt = PromptTemplate.from_template(prompt_template)

# Load model
model = AutoModelForCausalLM.from_pretrained('HUGGING FACE MODEL ID
    ', trust_remote_code=True)
tokenizer = AutoTokenizer.from_pretrained('HUGGING FACE MODEL ID
pipeline = transformers.pipeline(
```

```
    model=model,
    tokenizer=tokenizer,
    task="text-generation",
    eos_token_id=tokenizer.eos_token_id,
    pad_token_id=tokenizer.eos_token_id,
    repetition_penalty=1.1,
    return_full_text=True,
    max_new_tokens= 1000
)

llm = HuggingFacePipeline(pipeline=pipeline)
llm_chain = LLMChain(llm=llm, prompt=prompt)')

# Load abstracts
loader = PyPDFLoader("ABSTRACT PDF URL")
abstract_load = loader.load()

stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
   document_variable_name = "abstract")
answers = []

# Run model on abstracts
for i in range(47):
  abstract = [(abstract_load[i])]
  stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
   document_variable_name = "abstract")
  summary = stuff_chain.run(abstract)
  answers.append(summary)
  print('Summary',i+1,summary)

# Bert Score
pip install bert-score
from bert_score import score
P, R, F1 = score(cands=answers, refs=correct_summaries_red, lang="
   en", verbose=True)
average_precision = P.mean().item()
average_recall = R.mean().item()
average_f1 = F1.mean().item()

# Rouge score
pip install rouge_score
rouge = evaluate.load('rouge')
results = rouge.compute(predictions=answers, references=
   correct_summaries)
print(results)

```

```
# Perplexity score
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

perplexity_scores = []

for answer in answers:
    inputs = tokenizer(answer, truncation=True, padding=True,
    max_length=model.config.max_position_embeddings, return_tensors=
    "pt")
    with torch.no_grad():
        outputs = model(**inputs, labels=inputs["input_ids"])
        loss = outputs.loss
    ppl = torch.exp(loss)
    perplexity_scores.append(ppl.item())

average_perplexity = sum(perplexity_scores) / len(perplexity_scores
    )
```

Translation script outline

```

pip install --upgrade --quiet  langchain-openai tiktoken chromadb
    langchain accelerate transformers
pip install evaluate
pip install rouge_score
pip install pypdf

import transformers
import torch
import numpy as np
import evaluate

from langchain import PromptTemplate, LLMChain
from langchain.llms import HuggingFacePipeline
from langchain.chains.summarize import load_summarize_chain
from langchain.chains.combine_documents.stuff import
    StuffDocumentsChain
from langchain_community.document_loaders import PyPDFLoader,
    TextLoader
from langchain.text_splitter import CharacterTextSplitter
from evaluate import load
from transformers import AutoModelForCausalLM, AutoTokenizer,
    AutoConfig, BitsAndBytesConfig

import os
```

```
os.environ['HF_TOKEN'] = 'PERSONAL HUGGING FACE TOKEN KEY'

# Data
correct_translations = [LABELED TRANSLATIONS]

# Prompting
prompt_template = "YOUR PROMPT HERE"
prompt = PromptTemplate.from_template(prompt_template)

# Load the model
model = AutoModelForCausalLM.from_pretrained('HUGGING FACE MODEL ID
    ', trust_remote_code=True)
tokenizer = AutoTokenizer.from_pretrained('HUGGING FACE MODEL ID'
pipeline = transformers.pipeline(
    model=model,
    tokenizer=tokenizer,
    task="text-generation",
    eos_token_id=tokenizer.eos_token_id,
    pad_token_id=tokenizer.eos_token_id,
    repetition_penalty=1.1,
    return_full_text=True,
    max_new_tokens= 1000
)
llm = HuggingFacePipeline(pipeline=pipeline)
llm_chain = LLMChain(llm=llm, prompt=prompt)

# Load the abstracts
loader = PyPDFLoader("ABSTRACT PDF URL")
abstract_load = loader.load()
stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
    document_variable_name = "abstract")
answers = []

for i in range(0,25):
  abstract = [(abstract_load[i])]
  stuff_chain = StuffDocumentsChain(llm_chain=llm_chain,
    document_variable_name = "abstract")
  translation = stuff_chain.run(abstract)
  answers.append(translation)
  print('Translation',i+1,translation)

# BLEU score
bleu = evaluate.load('bleu')
bleu_results = bleu.compute(predictions=answers, references=
    correct_translations, max_order = 2)

```

```
64 # Perplexity
65 if tokenizer.pad_token is None:
66     tokenizer.pad_token = tokenizer.eos_token
67 perplexity_scores = []
68 for answer in answers:
69     inputs = tokenizer(answer, truncation=True, padding=True,
       max_length=model.config.max_position_embeddings, return_tensors=
       "pt")
70     with torch.no_grad():
71         outputs = model(**inputs, labels=inputs["input_ids"])
72         loss = outputs.loss
73     ppl = torch.exp(loss)
74     perplexity_scores.append(ppl.item())
75 average_perplexity = sum(perplexity_scores) / len(perplexity_scores
       )
```

Appendix C

Prompts for all three tasks

Prompt for Classification

```
"The following is a document:  "abstract"
Based on this document, please identify the main themes.
The ONLY themes you are allowed to choose from are:
- Archaeology
- Architecture
- Artificial Intelligence
- Biology
- Business
- Chemistry
- Civil Engineering
- Climate Change
- Communication
- Cultural Studies
- Design
- Economics
- Education
- Electrical Engineering
- Energy Systems
- Entrepreneurship
- Fashion
- Finance
- Gender Equality
- Gender Studies
- Geography
```

```
- Global Health
- Healthcare
- Healthcare Systems
- Human Resources
- Human Rights
- Humanities
- Industrial Management
- Information Technology
- Integration
- Language
- Law
- Literature
- Machine Learning
- Marketing
- Mathematics
- Mechanical Engineering
- Medicine
- Military Science
- Music
- Organizational Management
- Physics
- Political Science
- Project Management
- Psychology
- Religion
- Social Science
- Social Structures
- Social Work
- Sports
- Sustainability
- Technology
- Urban Planning
Please choose only a few themes from the list as the main themes
of the text.  ONLY choose themes from the list above.  Do not
come up with any other themes.
Theme:"
```

Prompt for Summarization

```
" The following is a document:  "abstract" Your task is to write
a concise summary of the document.  "
```

Prompt for Translation

" Följande är ett dokument: "abstract" Din uppgift är att översätta texten till svenska. Do not print the prompt. Only the translation."

Appendix D

User survey participants

In Table D.1 below, the participant information from the user survey is being presented.

Table D.1: Participant Information

Number	Age	Location	Sex	Academic Field
1	24	Sweden	Female	Natural science
2	25	Sweden	Female	Computer science and information technology
3	26	Sweden	Female	Natural science
4	26	Sweden	Female	Computer science and information technology
5	25	Sweden	Female	Computer science and information technology
6	27	Sweden	Female	Technology
7	27	Sweden	Female	Social science
8	25	Sweden	Female	Social science
9	54	Sweden	Female	Social science
10	25	Sweden	Female	Economy and business
11	26	Sweden	Female	Medicine and health studies
12	26	Sweden	Female	Computer science and information technology
13	26	Sweden	Female	Social science
14	28	Sweden	Female	Economy and business

User survey questions

Mänsklig evaluering av AI-modeller.

Denna undersökning syftar till att samla in återkoppling på prestandan hos olika AI-modeller använda för tre olika uppgifter:

1. Klassificering

2. Sammanfattning
3. Översättning från engelska till svenska.

Vänligen ranka modellernas svar från bäst till sämst enligt anvisningar längre ner.

Ditt abstract: THE PARTICIPANT'S ABSTRACT

1. Klassificering
Modellernas uppgift var att välja de ämnen som bäst klassificerar uppsatsen, givet abstractet. Vänligen läs igenom alla modellers svar och ranka vilken du tyckte svarade bäst.

Svar Modell 1: MODEL 1 OUTPUT
Svar Modell 2: MODEL 2 OUTPUT
Svar Modell 3: MODEL 3 OUTPUT
Svar Modell 4: MODEL 4 OUTPUT

Vänligen ranka vilken av de fyra modellernas svar från bäst till sämst. Observera att två svar kan dela ranking. *RANKING*

Är det något du vill kommentera angående dessa svar? Reagerade du på något särskilt som var konstigt?

2. Summering
Modellernas uppgift var här att skriva en kortfattad summering av abstractet. Tänk på att både språket ska vara grammatiskt korrekt och att innehållet ska vara relevant. Vänligen läs igenom alla modellers svar och ranka vilken du tyckte svarade bäst.

Svar Modell 1: MODEL 1 OUTPUT
Svar Modell 2: MODEL 2 OUTPUT
Svar Modell 3: MODEL 3 OUTPUT
Svar Modell 4: MODEL 4 OUTPUT

Vänligen ranka vilken av de fyra modellernas svar från bäst till sämst. Observera att två svar kan dela ranking. *RANKING*

Är det något du vill kommentera angående dessa svar? Reagerade du på något särskilt som var konstigt?

3. Översättning
Modellernas uppgift var här att översätta abstractet från engelska till svenska . Tänk på att både språket ska vara grammatiskt korrekt och att innehållet ska vara relevant.

Vänligen notera att för översättningsuppgiften har vi jämfört fem olika modeller istället för fyra.

Svar Modell 1: MODEL 1 OUTPUT
Svar Modell 2: MODEL 2 OUTPUT
Svar Modell 3: MODEL 3 OUTPUT
Svar Modell 4: MODEL 4 OUTPUT
Svar Modell 5: MODEL 5 OUTPUT

Vänligen ranka vilken av de fyra modellernas svar från bäst till sämst. Observera att två svar kan dela ranking. *RANKING*

Är det något du vill kommentera angående dessa svar? Reagerade du på något särskilt som var konstigt?

DEPARTMENT OF COMPUTER SCIENCE | LUNDS TEKNISKA HÖGSKOLA | PRESENTED 2024-05-29

MASTER'S THESIS Building a Framework for Evaluation of Small Language Models: A Study of NLP Task Performances and Swedish Comprehension
STUDENT Amanda Axelsson, Allis Rodvaldr
SUPERVISORS Maj Stenmark (LTH), Henrik Tingström (Scholaro)
EXAMINER Elin A. Topp (LTH)

Small language models are competing with Open AI's GPT-4 on NLP tasks

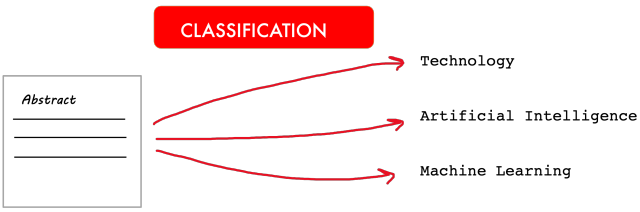
POPULAR SCIENTIFIC SUMMARY **Amanda Axelsson, Allis Rodvaldr**

Large language models (LLMs) are a type of AI model, used for natural language processing (NLP) tasks, such as classification, summarization, and translation from English to Swedish. However, one major problem with the implementation of LLMs is the extensive computational resources needed. It appears small language models (SLMs) can serve as a more cost-effective alternative for applications using NLP.

The field of SLMs is rapidly expanding. Their potential applications and how they compare to LLMs are only beginning to be explored. There exist a lot of different SLMs, and many of them are available open-source at HuggingFace.co. Some of these are instruction-tuned, meaning they are trained to follow instructions, similar to the well-known LLM GPT-4. We compared the performance of five of these SLMs on the NLP tasks classification, summarization, and translation, to each other and to GPT-4.

The models GPT-SW3, Mistral, OpenOcra, StableBeluga, and Qwen were tested on the mentioned NLP tasks. One possible application for these models is to match students' master's theses with appropriate scholarships. With this application in mind, the test data consisted of 300 master's thesis abstracts. The dataset was created and labeled, and is now available on Hugging Face.

An evaluation framework was built, and a range of metrics were employed. Some of the most commonly used metrics for evaluation of LLMs were applied, comparing the outputs from the models with the correct labels. Moreover, a user survey was carried out, in order to get human input as well. The results from the automatic evaluation sometimes differed from the user survey, which is an interesting insight. For example, Mistral received a higher automatic score than Qwen, but the user survey ranked Qwen higher.



In other cases, the automatic evaluation results aligned with the user survey. For translation from English to Swedish, GPT-SW3 was superior. This outcome was expected, since GPT-SW3 is the only model out of the five trained on a large Swedish dataset.

Comments from the user survey mentioned that some of the models produced unexpected outputs. For example, Qwen sometimes produced Chinese tokens when trying to translate from English to Swedish. Chinese is one of the languages that Qwen is trained on, which is why it probably confused a foreign language with another. Moreover, the user survey participants highlighted the difficulty of translating complex academic words.

The insights from the study were that if the correct model, prompt, and task are combined, SLMs can indeed compete with LLMs to a certain extent. Most importantly, a variety of automatic and human evaluation metrics are necessary for a full assessment of an SLM's performance.