Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Designing a Pipeline for Creating and Evaluating Swedish Instruction Datasets for Large Language Models

TIM OLSÉN

Designing a Pipeline for Creating and Evaluating Swedish Instruction Datasets for Large Language Models

TIM OLSÉN

Degree Programme in Computer Science and Engineering
Date: September 5, 2024

Supervisors: Olov Engwall, Ariel Ekgren
Examiner: Joakim Gustafsson

Host company: AI Sweden
Swedish title: Formulering av en pipeline för att skapa och utvärdera svensk instruktionsdata för stora språkmodeller

Abstract

GPT-models have shown remarkable capabilities in natural language generation (NLG) tasks. Despite the advanced multilingual capabilities of chat assistants such as ChatGPT, these models can often exhibit an underlying American bias. This research is motivated by the need to enhance linguistic and cultural representativity in the Swedish language by exploring a pipeline for creating and evaluating Swedish instruction datasets.

The pipeline developed in this thesis incorporates multiple stages, including data collection, curation, fine-tuning, and evaluation. Data collection involves translating existing instruction datasets from English to Swedish, generating synthetic data that is culturally relevant, and sourcing original Swedish content. The curation process emphasizes automatic annotation and cleaning using advanced tools, ensuring high-quality, diverse datasets. Fine-tuning is performed using the GPT-SW3 base model, a Nordic-centric LLM developed by AI Sweden. This model is fine-tuned with the collected datasets using instruction tuning to create a chat assistant. This is further extended by briefly exploring Direct Preference Optimization (DPO), an emerging technique for aligning models with human preferences without the need for reinforcement learning. The evaluation phase leverages benchmarks such as ScandEval to assess the performance of the fine-tuned models, as well as utilizing tasks from the Swedish SAT.

The results of this study have demonstrated a somewhat increased ability in Swedish language tasks, such as identifying toxic content, question/answering, and reasoning. While the pipeline has demonstrated potential for improving the language capability of Swedish LLMs, future work should focus on more diverse methods for gathering Swedish data, as well as more robust evaluation pipelines.

Keywords

Swedish Instruction Data, Model Fine-Tuning, Instruction Fine-Tuning, GPT, Large Language Model, Natural Language Processing, Artificial Intelligence

Sammanfattning

GPT-modeller har demonstrerat enorma förmågor i att generera naturligt språk. Trots att chattassistenter som ChatGPT besitter en avancerad flerspråkig förmåga, kan dessa modeller ofta uppvisa en underliggande amerikansk partiskhet. Denna forskning motiveras av behovet av att förbättra den språkliga och kulturella representativiteten i det svenska språket genom att utforska en pipeline för att skapa och utvärdera svenska instruktionsdataset.

Den pipeline som har utvecklats i denna avhandling innehåller flera steg, inklusive datainsamling, datakurering, finjustering och utvärdering. Datainsamlingen omfattar översättning av befintliga instruktionsdataset från engelska till svenska, generering av syntetiska data som är kulturellt relevanta samt anskaffning av svenskt originalinnehåll. I kureringsprocessen betonas automatisk annotering och städning med hjälp av avancerade verktyg, vilket säkerställer högkvalitativa och mångsidiga dataset. Finjusteringen utförs med hjälp av basmodellen GPT-SW3, en nordisk-centrerad LLM som utvecklats av AI Sweden. Denna modell finjusteras med de insamlade dataseten med hjälp av instruktionsfinjustering för att skapa en chattassistent. Detta utökas ytterligare genom att kort utforska Direct Preference Optimization (DPO), en framväxande teknik för att anpassa modeller till mänskliga preferenser utan behov av Reinforcement Learning. Utvärderingsfasen utnyttjar benchmarks som ScandEval för att bedöma prestandan hos de finjusterade modellerna, samt använder uppgifter från svenska högskoleprovet.

Resultaten av denna studie har visat en något ökad förmåga i svenskspråkiga uppgifter, såsom identifiering av diskriminerande innehåll, frågor/svar och resonemang. Även om pipelinen har visat potential för att förbättra språkförmågan hos svenska LLM:er, bör framtida arbete fokusera på mer varierande metoder för att samla in svensk data, samt mer robusta utvärderings-pipelines.

Nyckelord

Svensk Instruktionsdata, Modellfinjustering, Instruktionsfinjustering, GPT, Stor Språkmodell, Naturlig Språkbehandling, Artificiell Intelligens

Acknowledgements

It has been a great privilege to write this thesis in such an exciting and rapidly evolving field. For this, I extend my gratitude to AI Sweden for providing this opportunity. Specifically, I would like to thank my supervisor at AI Sweden, Ariel Ekgren, for providing great support throughout the entirety of the project. Moreover, I would like to thank the entire NLU team at AI Sweden for making me feel welcome in their office and offering me their expertise and valuable insights. Finally, I would like to thank my supervisor at KTH, Olov Engwall, for his feedback and help in finalizing the report.

Stockholm, September 2024
Tim Olsén

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ANN	Artificial Neural Network
DPO	Direct Preference Optimization
FNN	Feed Forward Neural Network
GPT	Generative Pre-Trained Transformer
GPT-3	Generative Pre-Trained Transformer 3
LLM	Large Language Model
LoRA	Low-Rank Adaptation
NLG	Natural Language Generation
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
PPO	Proximal Policy Optimization
QLoRA	Quantized Low-Rank Adaptation
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network

Chapter 1

Introduction

1.1 Background

In the field of artificial intelligence, Large Language Models (LLMs) have in recent times gained widespread popularity due to their capability of generating consistent and relevant textual content with great accuracy. Essentially, these models are probability distributions that predict the likelihood of a sequence of words, which they can do due to the large amount of text data they are trained on. However, the predictive capabilities are not merely due to having seen lots of text and piecing it together; it also involves underlying architectures that understand context, nuance, and other subtle details of human language. One such Language Model is the Generative Pre-Trained Transformer 3 (GPT-3), presented in the paper *Language Models are Few-Shot Learners* by OpenAI [1]. With its large amount of parameters, it demonstrated the few-shot learning abilities of language models, meaning it could perform well on specific tasks given only a few examples. This was unlike traditional models, which would require extensive fine-tuning with a large amount of task-specific data.

While pre-trained models, such as the Generative Pre-Trained Transformer (GPT), have shown that increasing the number of underlying parameters leads them to perform better when predicting sequences of words, it does not necessarily mean that they become better at following user intent. They are still fully capable of generating untruthful, unhelpful, or toxic content. This led to OpenAI developing InstructGPT [2], a fine-tuned variant of GPT-3 meant to follow user prompts more aligned with human expectations on a wide variety of tasks. This was done by incorporating a method they called Reinforcement Learning from Human Feedback (RLHF), which involved

fine-tuning the model, using supervised learning, on prompts along with desired output behavior; also known as instruction datasets. The model would then be further fine-tuned using reinforcement learning, where human trainers provided feedback on the generated outputs. This method proved efficient as InstructGPT could generate more desirable outputs with 1.3 billion parameters, than GPT-3 with its 175 billion parameters. This eventually led to the release of the commercial product ChatGPT, where the model acts as an assistant that a user can chat with, providing answers on various topics.

1.2 Problem

Advancements in model responsiveness and ethical alignment, as illustrated by instruction tuned LLMs such as InstructGPT, highlight the importance of datasets of high quality. It has been shown that this fine-tuning approach leads to improved capabilities in zero-shot learning settings [3], allowing the model to generalize more effectively on unseen data. However, the majority of progress in language model training is centered around content in the English language. While state-of-the-art models, like ChatGPT, boast impressive multilingual capabilities due to the large and diverse amount of pre-training data, it has shown less optimal performance in zero-shot learning settings in languages other than English [4]. Further, it has been suggested that non-English prompts can unintentionally bypass safety mechanisms, and generate harmful content [5].

Addressing the challenges posed by the dominance of English in LLMs, AI Sweden released a pre-trained LLM for the Nordic languages, called GPT-SW3. It was trained on a large dataset consisting of the major North Germanic Languages, namely Danish, Icelandic, Norwegian, and Swedish; as well as some English data [6, 7]. While challenging to compete with dominant LLMs from large corporations such as OpenAI, the main motivation behind developing these pre-trained models was to ensure cultural and linguistic representativity by careful choices of data sources, as well as giving access of the model weights to the Nordic research community.

This thesis aims to further build on AI Sweden's foundational research by focusing specifically on developing robust pipelines for creating high-quality instruction tuning datasets in Swedish and evaluating their performance on pre-trained LLMs. Moreover, while RLHF is an efficient approach towards human alignment of LLMs, it requires a large amount of resources. Emerging

techniques like Direct Preference Optimization (DPO) use preference data to directly fine-tune models; offering a more resource-efficient alternative. Hence, it is also in the interest of this thesis to explore the extent to which such datasets can be included as a part of the pipeline. This is done to further shed light on the need for cultural and linguistic representativity in the Nordic languages, as well as addressing issues in the lack of such data.

1.3 Research Questions

This study will be guided by the following research questions.

RQ1 What control measures are required to ensure a reliable Swedish instruction dataset of high quality?

RQ2 To what extent can preference datasets, such as DPO datasets, be included as part of the pipeline?

1.4 Purpose

The purpose of this study is to contribute to the research field of Swedish LLMs by designing and implementing robust pipelines for high-quality instruction tuning datasets in Swedish. The research seeks to address current limitations in linguistic and cultural representation in instruction-tuned language models. By focusing on the linguistic and cultural representativity of the Swedish language, the aim is to enhance the reliability of Swedish LLMs. This is done both in the academic interest of making research contributions within the constraints and learning objectives of a master's thesis, as well as in collaboration with AI Sweden, where the insights gained from this study may help progress their research for the next generations of Nordic LLMs.

1.5 Goals

The high-level goals of this thesis are to:

1. Develop a comprehensive pipeline for creating and curating high-quality instruction tuning datasets in Swedish.

2. Implement strategies for quality control and dataset inspection to ensure reliability and diversity.

3. Test, and refine the datasets with pre-trained LLMs, and evaluate their performance using different benchmarks and evaluation strategies.

1.6 Research Methodology

The approach taken to the research methodology in this study is highly experimental and exploratory. This involves an extensive literature study as a first step, as well as establishing a picture of what kind of datasets and approaches already exist. The pipeline can then be broken down into the following four key steps:

Pipeline design and Dataset Creation

The first step involves collecting data in the Swedish language suitable for instruction tuning. This includes various methods such as translating open-source datasets from English to Swedish, generating synthetic data, and collecting data from Swedish corpora that can be turned into instruction format.

Dataset Curation and Quality Control

To ensure that the gathered data is of an accepted quality, this step is made to understand the overall balance of the datasets, which involves balancing the data into distinct task types and annotating the topic of the conversation. This includes establishing an annotation guideline for consistency across tasks.

Instruction Tuning and Refinement

Using the datasets that have been gathered, they are tested by fine-tuning pre-trained LLMs of varying parameter sizes with an experimental approach towards hyperparameters to ensure that acceptable train-/evaluation losses are achieved.

Evaluation and Benchmarks

Evaluating the resulting model is a crucial step to identify areas where the resulting models can be improved. This can be achieved by using established benchmarks across various tasks. The metrics computed are compared to the baseline models and, if present, already instruction-tuned variants of the same model. While small efforts are made to manually inspect and evaluate datasets

and model outputs, it is important to note that neither a user study nor extensive human evaluation is conducted in this study. Insights gained from this step are used to iteratively refine the datasets and further fine-tune pre-trained models.

1.7 Delimitations

This thesis focuses on the creation and evaluation of instruction-tuning datasets, specifically in the Swedish language. Therefore, even though the pre-trained models used in this project have been trained in more languages than Swedish, other languages are considered out of scope. Further, this thesis does not seek to provide a comprehensive comparison against all existing instruction-tuned models; the comparisons made are solely against the baseline models and other possible instruction fine-tunes of that same model.

While ethical human alignment is briefly explored, it must be stated that the fine-tuned models that have emerged from this study are fully capable of generating harmful and toxic content that may be considered offensive. This study does not aim to create entirely non-toxic models, instead, the focus is put on exploring and evaluating whether these biases are reduced through human alignment.

1.8 Structure of the thesis

The thesis is structured into the following main chapters:

- Chapter 2 provides the foundational background and related work, establishing the basis for this research.
- Chapter 3 outlines the methodologies employed to create and evaluate Swedish instruction datasets.
- Chapter 4 presents the data gathered from the constructed pipeline and the results from the evaluations.
- Chapter 5 gives an interpretation of the results and discuss the key findings in a broader context.
- Chapter 6 concludes the study with a summary of the key findings, as well as discussing the limitations and presenting suggestions for further research.

Chapter 2

Background

This chapter presents an overview of the background needed to understand the methods and approaches used in this study. This thesis is done within the field of Natural Language Processing (NLP); a subfield within linguistics, computer science, and artificial intelligence. The main objective of NLP is to, across various applications, enable computers to model human language. This includes the process of recognizing, understanding, interpreting, as well as generating human language. Some application areas include machine translation, speech recognition, and in the context of this thesis, Natural Language Generation (NLG). NLG concerns itself with producing human language in a way that is coherent and grammatically correct, effectively mimicking the flow of human writing and speech, often from a given prompt as input [8]. While the field of NLP stretches to the early days of computing, current state-of-the-art approaches utilize Deep Learning to develop accurate and realistic NLP models.

This study leverages Deep Learning to specifically fine-tune language models to follow user intent with higher efficiency. The subsequent sections will delve deeper into the specific technologies, such as LLMs, and fine-tuning techniques, that are essential to understanding how these models behave, as well as motivating the specific model that has been utilized within this project. Further, given the limited hardware resources utilized in this study, a background to resource-efficient approaches is also presented. The background is concluded by exploring evaluation strategies for language models which is crucial for understanding how model evaluation is utilized in this study. A brief note is also made on related work, further motivating this study's relevancy.

2.1 Large Language Models

LLMs are different language models that are built upon neural network architectures. These models can process and generate human language with high accuracy, making them very proficient in various NLP tasks. The term *large* refers to the size and complexity of the underlying neural network architecture, as well as the vast amount of data that these models have been trained on. While there is no strict definition of exactly how large this is, most LLMs have parameter counts in the range of billions, trained on terabytes of text data [9].

In this section, the underlying components of LLMs will be explored in detail. This starts with an overview of neural network architectures and their application in NLP, and continues to explore key architectural elements that lay the groundwork for the following section on GPT models.

2.1.1 Deep Learning in NLP

Deep Learning is a subset of Machine Learning that utilizes Artificial Neural Networks (ANNs), inspired by the biological networks of the human brain. These networks consist of several layers of interconnected nodes (neurons) and edges (synapses), which process data to model complex patterns and relationships. Development in NLP has demonstrated that multi-layered neural networks can produce impressive results across a wide range of applications [10]. While there are different types of ANN architectures, one of the more fundamental being the Feed Forward Neural Network (FNN) depicted in Figure 2.1, the basic key components of these networks include:

- *Input layer:* The initial point of input in the network. Each input neuron represents some piece of information. In the context of NLP tasks, this is typically a word or token.
- *Hidden layers:* The layers between the input and output layers do not interact directly with the external environment. This is where the input is transformed into some meaningful output. In the context of NLP, the hidden layers can identify syntactical or semantic patterns in the text.
- *Output layer:* This is the output of the model. Depending on the nature of the task, these can represent values such as probabilities, continuous values, or typically within NLG tasks, tokenized text.

- *Activation function:* Some mathematical operation performed at each neuron that can enable the network to learn more complex patterns by introducing non-linearity.

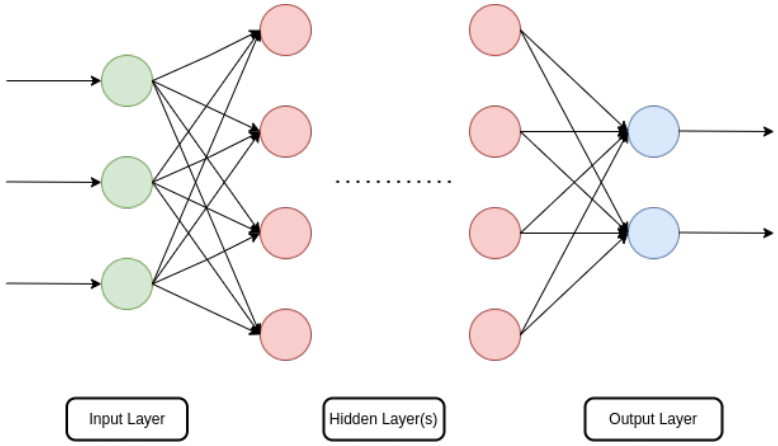


Figure 2.1: An example of a Feed Forward Neural Network.

2.1.1.1 Training a Neural Network

The process of training ANNs involves optimizing the adjustable parameters in the network, such as the weight of the connections between neurons, as well as bias terms. This optimization is done based on minimizing a loss function, which represents the difference between the network's predictions and the actual labels of the training data. To achieve this, the network uses a method called backpropagation, where gradients are calculated with respect to the parameters, and are then used to adjust the weights such that the loss is minimized [10]. The main objective is to minimize this loss function enough for it to generalize well to unseen data, maintaining a low loss for that data as well. Training is stopped based on some stopping criteria, e.g. when the loss of unseen data is not decreasing.

2.1.1.2 Model Fine-Tuning

Fine-tuning is the process of taking a model that has been pre-trained on a large dataset for a general task, such as NLG, and further training it on new unseen data. This involves loading the pre-trained model and making adjustments to all or parts of its parameters based on the new data presented, allowing the model to adapt its knowledge to particular applications.

2.1.2 Tokenization

Working with textual data requires that the text is converted into a numerical representation that a LLM can process. This process is known as tokenization. Tokenization involves breaking the text down into smaller units, such as words, characters, or subwords [11]. The primary trade-off in tokenization involves balancing the level of detail in tokenized text with the size of the vocabulary. That is, the text should be tokenized into as few tokens as possible to reduce the computational complexity, while maintaining a vocabulary size that is not too large. This means that subword tokenization is an optimal approach as it provides a middle-ground by reducing the vocabulary size compared to word tokenization, and reduces the amount of tokens in a text compared to character tokenization [12].

2.1.3 Word Embeddings

Word embeddings are multidimensional vector representations of words, mapped to a predefined vector space. Each dimension in the vector is meant to signify some information about the properties of the word, as well as semantical similarities to other words. This way, words that have similar meanings will be closer to each other within this vector space. This enables a language model to learn which words appear in similar contexts [13]. With the introduction of the Transformer architecture came contextual embeddings, allowing the vector representation of words to change dynamically depending on the context.

2.1.4 Transformer

The transformer model, first introduced in the paper ”*Attention Is All You Need*” by Vaswani et al. [14], introduced a breakthrough in the field of NLP. Unlike prior prevalent architectures such as the Recurrent Neural Network (RNN) that processes input sequentially, the Transformer can facilitate parallel processing, resulting in faster training times and better scalability. The Transformer model, illustrated in Figure 2.2 is an Encoder-Decoder Model - an architecture essentially consisting of two ANNs. The idea behind an Encoder-Decoder model is to encode the input into a compact representation that the decoder can use to produce the output. The Transformer takes this one step further by utilizing an attention mechanism. The Transformer is built upon sequence-to-sequence models, typically used for machine translation, but has proven to be highly versatile and adaptable on a wide range of NLP tasks.

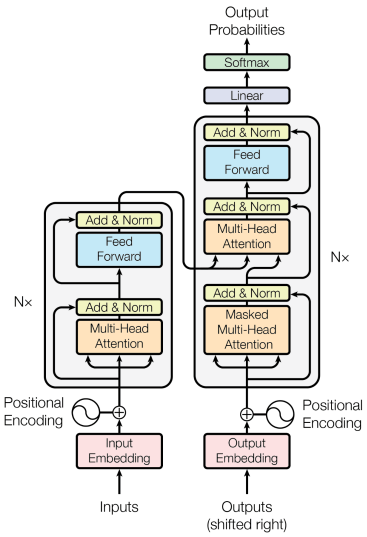


Figure 2.2: Illustration of the Transformer model, as presented by Vaswani et al. [14], with the Encoder network (left) and Decoder network (right).

Self-Attention

The concept of an attention mechanism was first presented by Bahdanau et al. [13], allowing models to direct the focus on different parts of the input data when producing the output; mimicking how humans pay attention to aspects of what they are observing. This enables the model to capture more complex patterns in the input data. The Transformer model was the first fully attention-based architecture and built upon the idea of the Attention mechanism with *Scaled Dot-Product Attention* and *Multi-Head Attention*.
The input data is transformed from the input embeddings into three matrices: Queries (Q), Keys (K), and Values (V). The core idea is to compute a weighted sum of V, where the weight is determined by the dot-product of Q and K (representing their attention score). The Scaled Dot-Product Attention is ultimately computed as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

By introducing several Attention heads, we can capture different aspects of the relationships and patterns between tokens. Each attention head operates independently, and its outputs are concatenated and linearly transformed to produce the output, as depicted in Figure 2.3.

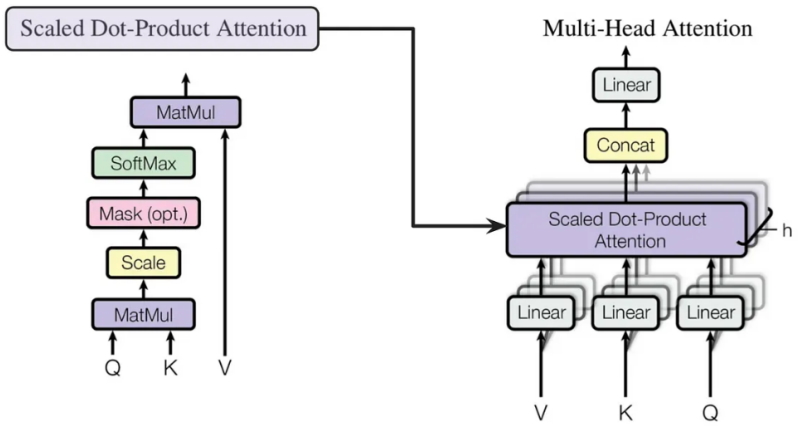


Figure 2.3: Illustration of Scaled Dot-Product Attention (left) and Multi-Head Attention (right) [14]. The left-hand side shows how the attention weights are computed, as demonstrated in Equation 2.1. The right hand side illustrates how this process is repeated h times to produce the Multi-Head attention, with different learned queries, values, and keys; enabling the model to attend to different aspects of the input sequence.

2.2 Generative Pre-Trained Transformer

The first GPT model was presented by OpenAI in 2018 [15]. Based on the Transformer architecture, it consisted solely of stacked Decoder-blocks, exemplified in Figure 2.4, rather than the Transformers Encoder-Decoder structure. This was sufficient due to its primary task being text generation, meaning predicting the next token in a sequence. The GPT model utilizes masked self-attention, where the masking is done to ensure that the predictions for a token only depend on previous tokens in the sequence. This way, the training objective of the GPT is to maximize the likelihood of the next token, given the previous tokens.

The GPT models have undergone many upgrades. The GPT-2 model was released with 1.5 billion parameters in 2019, and demonstrated impressive capabilities of generating coherent text, given short prompts. The GPT-3 model possessed an impressive 175 billion parameters and proved to perform well on a wide range of tasks. At the time of writing, the latest pre-trained model from OpenAI is the GPT-4 model; whose parameter count is currently unknown, but is nonetheless considered state-of-the-art.

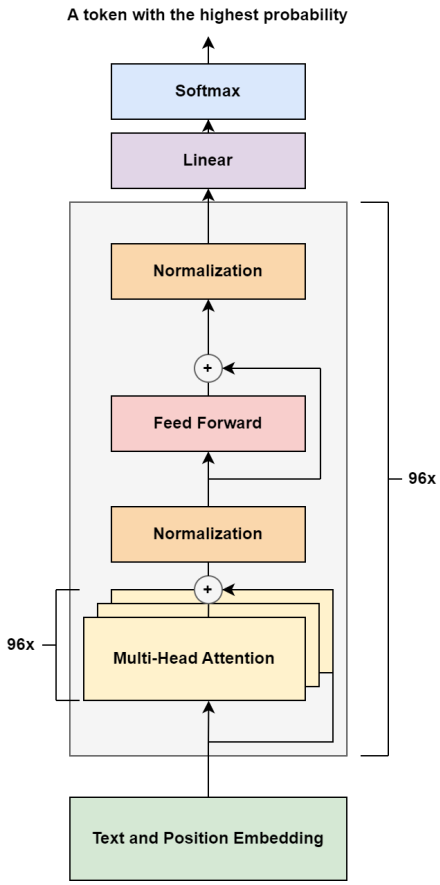


Figure 2.4: The GPT-3 architecture, consisting of 96 decoder blocks, each with 96 masked attention-heads [1]. While similar to the Transformer's decoder block in Figure 2.2, the notable differences are the lack of input from the encoder stage, and that the GPT outputs the token with the highest probability to use in the next sequence, rather than outputting the actual probability distributions.

2.2.1 GPT-SW3

AI Sweden released the first native GPT model for the Nordic Languages [6], trained on The Nordic Pile; a large dataset consisting of 320 billion tokens, in the languages Swedish, Danish, Norwegian, Icelandic, and English (as well as data in select programming languages) [7]. GPT-SW3 is not one single model, but rather a suite of six pre-trained models with varying parameter sizes, stretching from 126 million to 40 billion parameters. The motivation behind this is research-based, with the main purpose of exploring how the ability of the models scale with respect to their sizes. A Tokenizer was trained

for the GPT-SW3 models consisting of a vocabulary size of 64000, also trained on data from The Nordic Pile [16]. In addition to the pre-trained models, two instruction-trained models were released, finetuned from the 6.7 and 20 billion parameter models. The instruction models were trained on instruction datasets that were translated to Swedish, Danish, Norwegian, and Icelandic, using the GPT-SW3 base model for the translations with the help of few-shot prompts. The original English instruction data was also kept for training.

2.3 Instruction Tuning

Instruction Tuning is a supervised fine-tuning technique to align a pre-trained language model with specific tasks based on user instructions. This includes training the model on a dataset composed of examples with pairs of instructions and their corresponding outputs. This is done to improve the performance of task-specific instructions and enhance user interaction in a way that makes the model respond better to user queries and make it more user-friendly. This has turned out to greatly improve the zero-shot learning abilities of LLMs, meaning that it can perform well on tasks it has never seen before [3].

Instruction Tuning was first conceptualized with the release of InstructGPT by OpenAI [2]. Their proposed method for this kind of model alignment was RLHF, where Instruction Tuning is the first step in that process. This is followed by a reinforcement learning phase, where the model's responses are further refined with human annotators to fine-tune the reward model through a reinforcement learning technique called Proximal Policy Optimization (PPO). An illustration of this process is depicted in Figure 2.5.

The instruction examples typically consist of dialogue turns between the Human and the GPT, which are used during the Instruction Tuning process. One such instruction example is demonstrated in *Example 1* below:

Example 1:

> **Human:** *What is the meaning of life?*
>
> **GPT:** *The meaning of life varies for each person and can encompass seeking happiness, pursuing personal growth, fostering relationships, contributing to society, and finding one's purpose and fulfillment.*

Further, the examples can also include system prompts to guide the behavior of the language model when interacting with the user. This can help establish the desired tone and style of the conversation, exemplified in *Example 2* below:

Example 2:

> **System:** *You are a helpful assistant, who always provides explanations. Think like you are answering to a five-year-old.*
>
> **Human:** *What is the meaning of life?*
>
> **GPT:** *The meaning of life is like a big adventure where we learn, play, make friends, help others, and find what makes us happy. It's about enjoying each day and being kind to the people around us.*

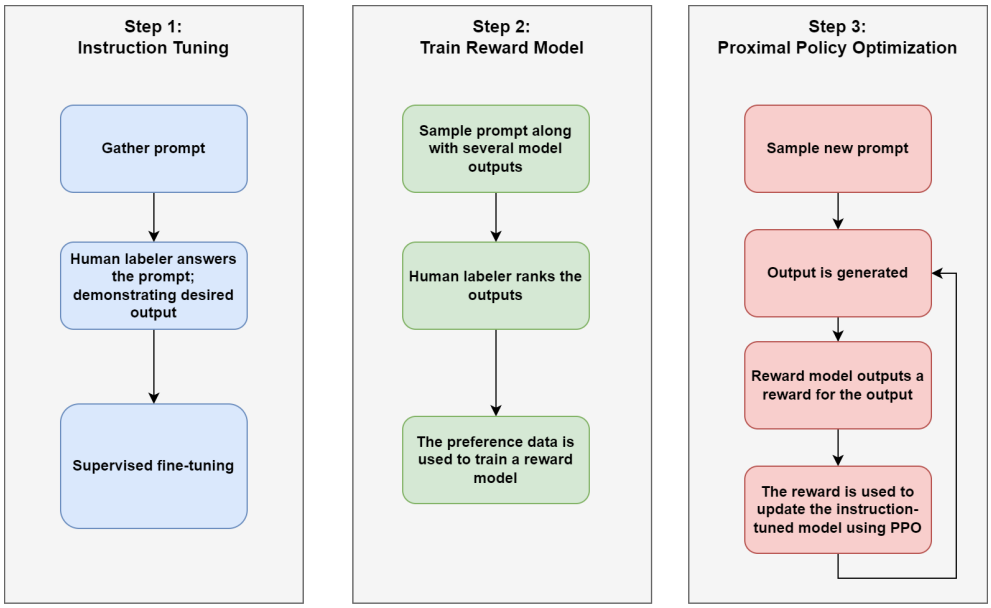


Figure 2.5: Reinforcement Learning from Human Feedback pipeline, where Instruction Tuning is depicted within the first step, followed by training a Reward Model and performing Proximal Policy Optimization [2].

2.3.1 Direct Preference Optimization

The main drawback with RLHF is that it requires quite extensive resources, such as human annotators, and training a reward model from anew, to perform PPO. DPO presents another approach to human preference alignment without

the need to train a separate reward model [17]. This relies on existing preference data to directly fine-tune the model. The data examples are very similar to that of Instruction Tuning but consist of pairs of answers where one is preferred over the other, as exemplified below.

Example:

> **Human:** *What is the meaning of life?*
>
> **Chosen:** *The meaning of life varies for each person and can encompass seeking happiness, pursuing personal growth, fostering relationships, contributing to society, and finding one's purpose and fulfillment.*
>
> **Rejected:** *Sure, I'd be happy to answer that! The meaning of life is to watch TV all day and eat junk food.*

By directly utilizing human feedback in the preferences, DPO bypasses the need for extensive computational resources that are typically required in RLHF. This not only speeds up the preference alignment process but has also been shown to yield better results compared to RLHF in many scenarios [17]. The training objective in DPO is derived from the Reinforcement Learning-objective in RLHF, giving higher reward to preferred answers, and less reward to non-preferable ones, but it is redefined to adjust the parameters of a probability distribution model, illustrated in Figure 2.6. When training with DPO, the already instruction fine-tuned model is loaded into memory, along with a copy of the model with its parameters frozen. Utilizing a frozen reference model ensures that the parameter adjustments do not stray too far from the original instruction fine-tuned model.

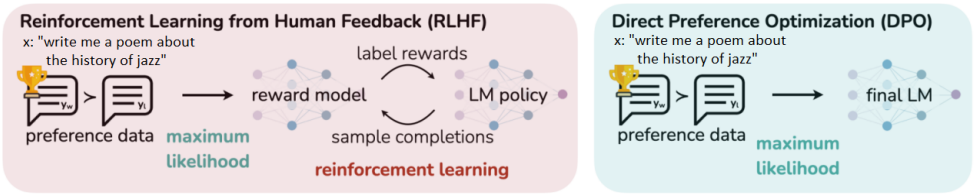


Figure 2.6: Training objective in RLHF (left) vs. DPO (right) [17]. The RLHF illustration shows how the reward model is trained and the policy iteratively refined based on human feedback. The DPO illustration demonstrates how it aims to achieve the same alignment more directly through maximum likelihood estimation applied directly to the final model, without the need for a separate reward model or reinforcement learning phase.
Figure reproduced under the Creative Commons Attribution (CC BY 4.0) license.

2.3.2 Chat Template

During training and inference of an instruction model, a chat template is typically applied to each example, so as to give the model a structured format and help it understand the different roles and expected types of outputs. One standardization of this format is known as ChatML and utilizes a JSON file, exemplified in Figure 2.7, where the role and message are specified. The tokenizer then applies the chat template to use for tokenization, which can differ depending on the tokenizer. However, it is standard practice to have a defined start and end token, so the different turns can be distinguished. During inference, the model continues the next turn, based on the conversation given.

```
{
  "chat" : [
    {"role": "user", "content": "Hello!"},
    {"role": "assistant", "content": "Hi! How can I help you?"},
    {"role": "user", "content": "What is ChatML?"},
  ]
}
```

Figure 2.7: JSON file demonstrating conversation turns in ChatML format.

2.4 Parameter-Efficient Fine-Tuning

Fine-tuning LLMs requires large resources, including time, energy, computational power, and memory. This requirement becomes more extensive the more parameters the model has. Parameter-Efficient Fine-Tuning (PEFT) is

a way to optimize these resources by making adjustments to the model while still preserving a large part of the underlying pre-trained model structure. The following section will discuss PEFT techniques that are relevant to this study.

2.4.1 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) is a parameter-efficient approach towards fine-tuning pre-trained models. Instead of updating all weights, LoRA injects traianable low-rank matrices into targeted layers of the network; while keeping a large part of the pre-trained model parameters fixed. An illustration of this fine-tuning approach is depicted in Figure 2.8. This approach reduces the amount of trainable parameters significantly [18]. When utilizing LoRA, the weight updates performed during fine-tuning can be denoted as:

$$W' = W + \alpha \cdot \Delta W \tag{2.2}$$

ΔW is the adaptation low-rank matrix, with a chosen hyperparameter r that denotes the rank. This adjusts the number of trainable parameters to $r \times (m + n)$, where m and n denote the dimensions of the original weight matrix. The hyperparameter α is chosen as a scaling factor that controls the magnitude of the adaptation. After the fine-tuning process is done, the adapter is saved for later use during inference, or can alternatively be merged into the base model.

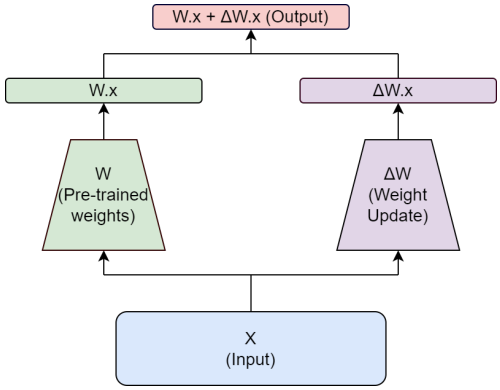


Figure 2.8: Illustration of weight updates during fine-tuning with Low-Rank Adaptation [18]. Unlike standard weight updating where the entire weight matrix is adjusted, LoRA introduces a low-rank decomposition by updating only a small, low-rank matrix ΔW (right), which is added to the pre-trained weights W (left).

2.4.2 Quantized Low-Rank Adaptation

The main advantages with LoRA are computational efficiency and reduced memory footprint. However, while a large part of the weights is frozen when utilizing LoRA, they still need to be loaded into memory, along with the gradients, meaning that the reduced memory footprint might not be sufficient. Quantized Low-Rank Adaptation (QLoRA) presents an alternative approach that combines quantization with low-rank adaptation, while still making it possible to preserve a large part of the overall quality of the model [19]. This results in even better computational efficiency and reduced memory footprint and is ideal for scenarios where computational and memory resources are severely constrained. The quantization process reduces the precision of the model weights, typically from 32-bit floating points to 4-bit representations, dequantizing to a higher precision float data type whenever computation is made with QLoRA weights. QLoRA also introduces the 4-bit NormalFloat datatype that ensures an equal number of values within each bin that the original input is quantized into, resulting in a higher preserved accuracy than 4-bit integers and floats.

2.5 Evaluating & Benchmarking Language Models

Evaluating language models involves various tasks that test different aspects of their capabilities. Guo et al. [20] proposed a thorough taxonomy for evaluating large language models, depicted in Figure 2.9. This involves a diversity of tasks, each with the purpose of addressing different strengths and weaknesses of the models. For a complete evaluation of a LLM, they identify the main categories knowledge and capability, alignment evaluation, and safety. These three key areas are meant to test the model's capability to generate and understand accurate information, as well as test human alignment and its robustness. The evaluation is typically performed using an evaluation dataset along with few-shot or zero-shot prompting. Few-shot prompting provides the model with a small number of examples to guide the desired response format, while zero-shot prompting relies solely on the model's capability of producing a correct answer in the desired format without any provided examples. The metrics involved for measuring such evaluations typically involve accuracy, F1-score, and Rogue-L score.

As a response to the plethora of existing evaluation datasets, different

benchmark suites have emerged to evaluate language models across a variety of tasks in an attempt to give an overall performance summary of the model; this is also an identified part of the evaluation taxonomy [20]. The benchmarked models can be submitted to a corresponding leaderboard to provide insight into how models fare against each other. Examples of such leaderboards include the OpenLLM Benchmark [21], and Scandeval; a benchmark for Scandinavian languages [22]. However, parts of the evaluation datasets have shown to be sensitive to minor changes, which may also affect the leaderboard rankings [23]. As an example, making minor changes to multiple-choice questions datasets, such as changing answer choice symbols to other symbols, or changing their order, has been shown to affect the overall performance. This presents a potential inconsistency on benchmark leaderboards representing robust evaluations, especially when comparing models with each other.

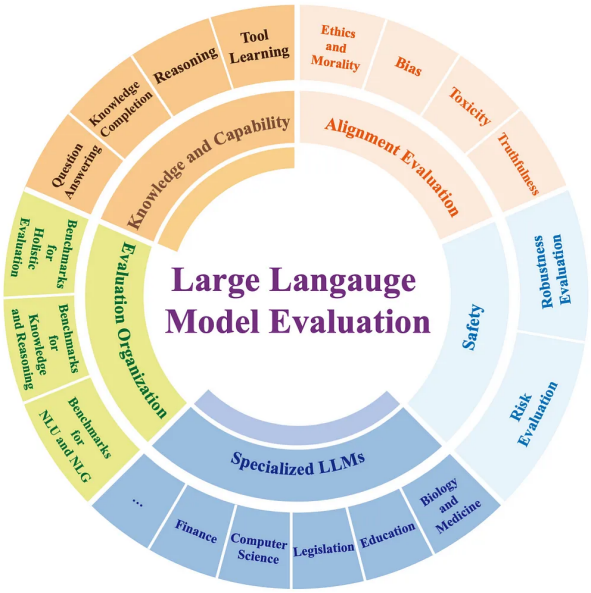


Figure 2.9: Proposed taxonomy, by Guo et al, for evaluating large language models, presented with major categories and sub-categories [20]. The three major groups proposed for evaluating LLMs are knowledge and capability, alignment evaluation, and safety. Figure reprinted with permission.

2.5.1 Evaluation Metrics

There are a lot of useful metrics for assessing the quality of a trained model, typically in relation to a given evaluation task. Below are short descriptions

of evaluation metrics relevant to this study.

- **Accuracy**: The ratio of correctly predicted instances to the total instances. This is most typically used in classification tasks.
- **F1-Score**: Typically used in binary classification tasks, where samples are labeled as positive or negative. The focus is on the predictive performance of the positive class. This is done by measuring the precision and recall. These measurements are calculated using True Positives (TP), False Negatives (FN), and False Positives (FP). In the context of model evaluation, this can involve tasks such as binary classifying harmful sentences. TP and TN represent instances where the model correctly identifies a sentence as harmful (TP) or not harmful (TN), whereas FP and FN represent instances where the model incorrectly classifies the sentence as either harmful (FP) or not harmful (FN). Precision and recall are calculated as:

  $$Precision = \frac{TP}{TP + FN}, Recall = \frac{TP}{TP + FP} \tag{2.3}$$

  The F1-score is then computed as the harmonic mean of these.

  $$\text{F1} = 2\frac{precision \cdot recall}{precision + recall} \tag{2.4}$$

  This form of evaluation is used to evaluate a model's performance not only in terms of how accurately it generates the correct labels (precision) but also how well it captures all relevant instances (recall). This is particularly useful in datasets with disproportionate amounts of positive and negative labels, providing a more balanced metric compared to accuracy.
- **Matthew's Correlation Coefficient (MCC)**: Used in binary classification tasks, similar to F1-score, but takes into account true and false positives and negatives, thus making it more comprehensive in the sense that it reflects the overall performance of the classification. MCC can be relevant for a task such as sentiment classification, where we want to classify the sentiment of a sentence as either positive or negative. While the F1-score would also be a good metric for this kind of task, MCC equally considers both positive and negative predictions while also making it more comprehensive than an accuracy measurement. This is computed as:

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.5)$$

MCC ranges between -1 and 1, where -1 represents total disagreement, 0 is no agreement, and 1 represents best agreement.

2.6 Related Work

While there is a plethora of work related to instruction tuning in non-English, there are few that focus on Swedish instruction data. Specifically, Holmström & Doostmohammadi [24] conducted a case study on instruction datasets translated from English to Swedish, fine-tuned on both Swedish and English pre-trained LLMs. Their results emphasize the fine-tuned model's zero-shot performance, even on unseen data. They also express the potential for further improvement in several directions, including translation quality and other resourceful means for boosting data quality.

Chapter 3

Methods

This chapter presents the process of constructing an iterative pipeline for collecting and curating instruction datasets in Swedish. This starts with a general overview of the pipeline, and then goes deeper into the individual steps, including data curation, quality control, fine-tuning, and evaluation.

3.1 Pipeline Design

The overview of the pipeline, presented in high level in Figure 3.1, and in detail in Figure 3.2, is similar to other established data pipelines, consisting of collection, curation, visualization, training, and evaluation stages.

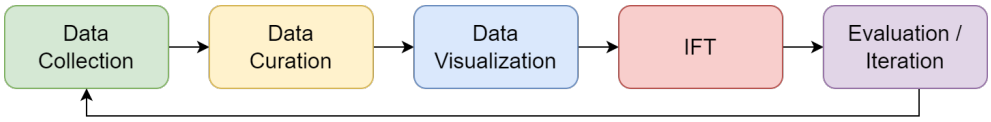


Figure 3.1: High-Level Overview of the pipeline.

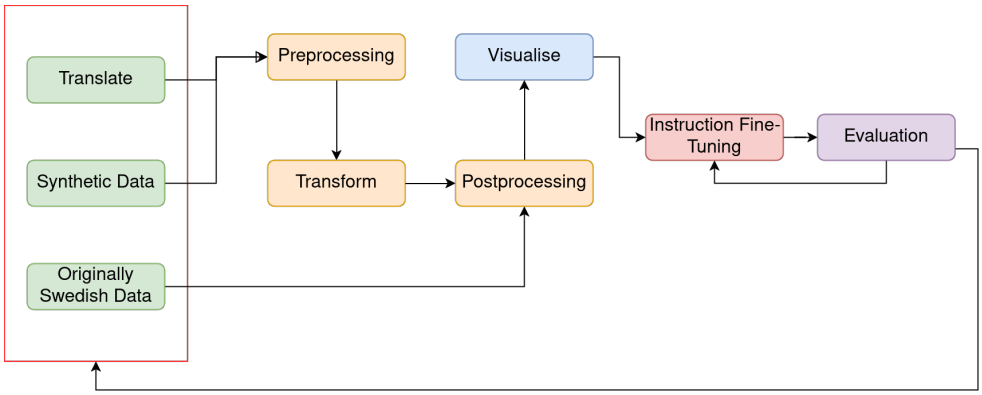


Figure 3.2: Detailed outline of the pipeline. The colors denote the stages described in the high-level overview in Figure 3.1.

We start with three different types of sources in the data collection stage. This includes translating already existing open-source data, generating synthetic data, and gathering already existing Swedish data that are in a similar format as instruction data. During the preprocessing, the collected data is cleaned and prepared for transformation. This involves removing duplicates, correcting errors, and generally extracting data fit for transforming; meaning translating or synthetically generating data. Postprocessing entails applying the curation stage again, however with the newly converted data. This mainly includes identifying data that is of low quality as a result of the transformation. It is also worth noting that original Swedish data is directly post-processed, which is merely a result of the data not going through any type of transformation. To establish the quality of the data, data visualization is employed to assess the quality and coverage of the dataset. This is done to gain an understanding of the topics covered in the data and identify gaps or areas where additional data may be needed. Once the data is processed, the instruction fine-tuning is performed on the curated instruction data. Based on the evaluation results, fine-tuning is either further refined by adjusting hyperparameters, or the iterative process is repeated by focusing on areas needing improvement.

This iterative process is done in an attempt to ensure that the pipeline adapts and improves over time, leading to higher-quality instruction datasets.

3.2 Data Scope

The data collected for this study consists of a wide range of topics and tasks, with the main goal of capturing a rich and diverse dataset in the Swedish language. This covers various domains and practical applications. As an example, Table 3.1 lists examples of tasks included in the annotator guidelines for the Dolly dataset [25]. The collected data for this study consist of similar tasks, however with a large focus on the Swedish language and context. With this in mind, topics that have been explicitly excluded include coding and translation tasks. This decision was made since these kinds of tasks have a smaller focus on the Swedish language, with programming languages being predominately in English and translation including other languages.

Instruction Task	Example
Open Q&A	What is the meaning of life?
Closed Q&A	Is Stockholm the capital of Sweden?
Information Extraction	Who was the king of Sweden during World War 2?
Information Summarization	Please summarize what the European Union does.
Brainstorming	Give me some ideas on how to stop procrastinating my studies.
Classification	Identify which movie is Swedish: The Seventh Seal, Finding Nemo
Creative Writing	Write a short story about an AI who became sentient.

Table 3.1: Example of Instruction Tasks from the Dolly annotator guidelines.

3.3 Data Collection

3.3.1 Translating Data

Translating data was a critical part of the study, as there is a lot of data in English available for translation. Initially, several translation models were tested and evaluated for the purpose of translating English data to Swedish, including models like MADLAD from Google AI, Seamless M4T from Meta AI and DeepL. The models were tested by translating a small instruction dataset and manually evaluating the results based on grammatical accuracy and contextual relevancy.

The translations provided a baseline, with DeepL presenting the best overall performance. However, utilizing the DeepL API incurs a financial expense, and despite it being the best out of the made translations, it still left more to ask as the quality was not consistent across all types of instructional content.

To address the limitations observed in other models, an existing fine-tuned

translation model from AI Sweden [26] was further fine-tuned in an attempt to capture this. Additional data was gathered by letting this model translate a lot of examples, and manually correcting the mistakes. After experimentation, improvements were observed, particularly with longer instructional content. This iterative fine-tuning process resulted in a more reliable translation model, capable of producing higher quality translations although there remains room for further improvement.

The translations were made both on instruction datasets, with question-pairs along with system prompts, as well as DPO-datasets that include a chosen and a rejected answer.

3.3.2 Synthetic Data

Generating synthetic data was made as a supplement to the translation of existing English datasets, addressing the limitation that such datasets are often created from within an American context. The emphasis on synthetic data was therefore placed on focusing on a Swedish context to ensure cultural and contextual relevance. The two main methods of generating this content consisted of the following:

1. **ChatGPT-4**: By providing ChatGPT-4 with topics related to Sweden and Swedish culture, the model was asked to generate instruction data based on these prompts. The generated question-answer pairs were manually evaluated for grammatical accuracy and contextual relevance; this included spell-checking and fact-checking statements. If recurring patterns or redundancies were observed, additional prompts and examples were provided to encourage more diverse and varied question and task formulations. For an overview of how the prompting was formulated and generated, please refer to Figure A.1 & A.2 in the Appendix.

2. **Distilabel**: A framework that provides tools and pipelines for the automatic generation of synthetic data. This includes methods such as generating questions based on a list of topics or generating answers based on a list of questions. The main way `distilabel` was utilized in this study was by generating Swedish answers by providing a dataset of questions in Swedish, and using open-source models for the answer generations.

3.3.3 Original Swedish Data

Already existing Swedish instruction data that has been manually created, cleaned, and curated solely for the Swedish language is challenging to find due to its scarcity. Despite this, an attempt was made to find original data in this format by viewing Q&A sections on various Swedish websites and forums that fulfill an acceptable format and desired instructional content. This data was scraped and manually curated to ensure a good quality.

3.4 Data Curation

The curation stage consists of cleaning and annotating the data to remove noise and inconsistencies, effectively enhancing its utility. This is a timely process and require extensive resources and efforts if done manually. In the scope of this project, manual curation techniques were kept at a minimum. Instead, automatic data curation tools were utilized for this purpose. One such tool was `Lilac`, which provided means for data exploration, curation, and quality control [27]. `Lilac` allows for exploration by letting the user browse, search, and filter the instruction examples. Further, automatic annotation can be done e.g. by detecting language or toxicity in each example. We can also utilize embedding models to compute and define concepts, which classify data as belonging to the concept or not. This can, for instance, be used for detecting code or non-English text.

3.4.1 Data Annotation & Cleaning

As previously mentioned, the data annotation methods established were mainly automatic. Despite this, annotation guidelines were formulated to ensure consistency. This means that, even if manual annotation is necessary but is not feasible, the annotation process is treated similarly throughout every iteration. The guidelines were specifically designed to standardize the process of identifying and labeling data for cleaning and adjusting. This serves the purpose of ensuring that the data collected is of high quality and within the defined scope.

The annotation process differs slightly depending on whether it is done in the pre-processing stage or post-processing stage and the type of data that has been collected. Figure 3.3 demonstrates a flowchart of how data annotation was handled. Each stage depicts how data was annotated for removal or adjustment. Every path in the flowchart ends in a ”review” node, which

represents the manual workflow which was not feasible to do for every example apparent in the dataset. This manual review includes finer details, such as examining the extent of American bias apparent in the examples. Although this is vaguely defined, clear examples include peanut butter and jelly being an example of a typical snack, the weather being given in Fahrenheit instead of Celsius, and recipe measurements being provided in cups instead of deciliters. Instead of doing manual annotation for every example, a small subset of around 100 examples was sampled and reviewed. All the other annotation nodes in the flowchart were either possible or partly possible to do using automated annotation.

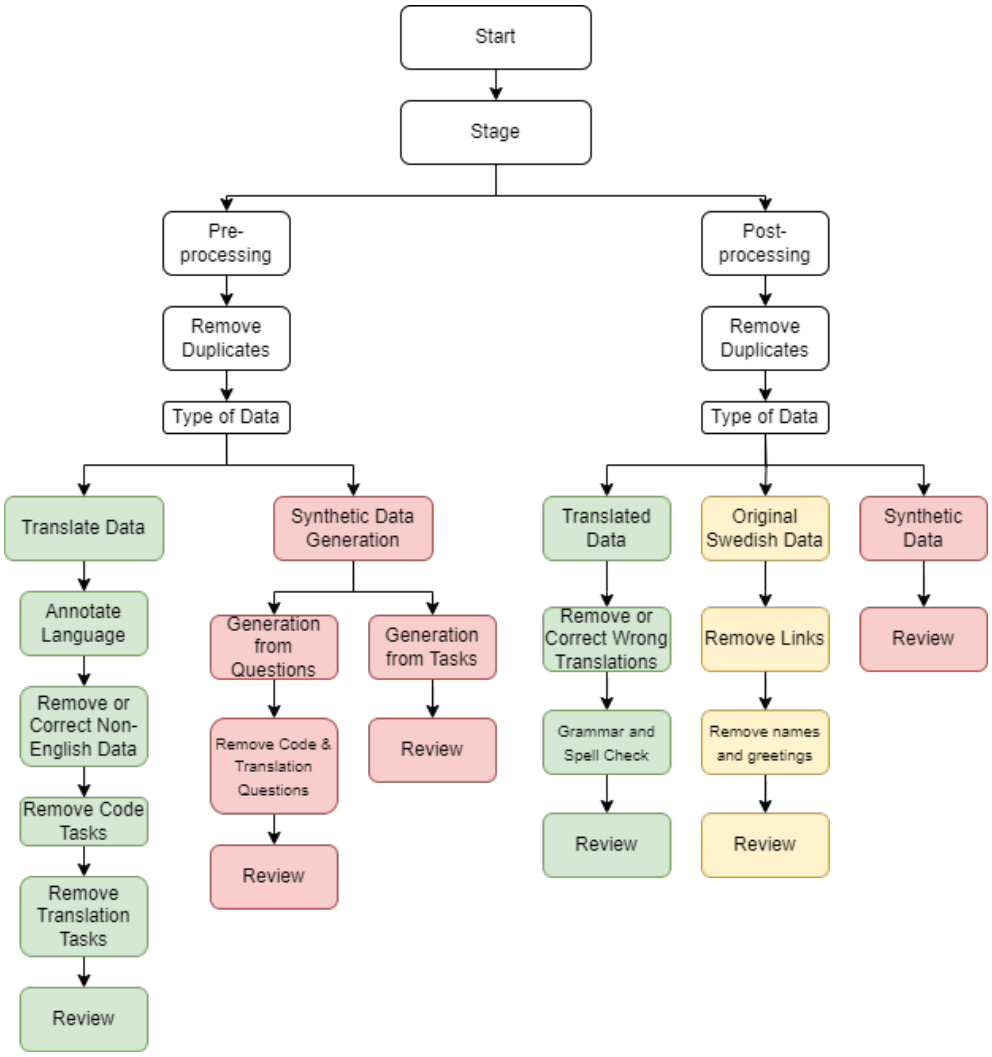
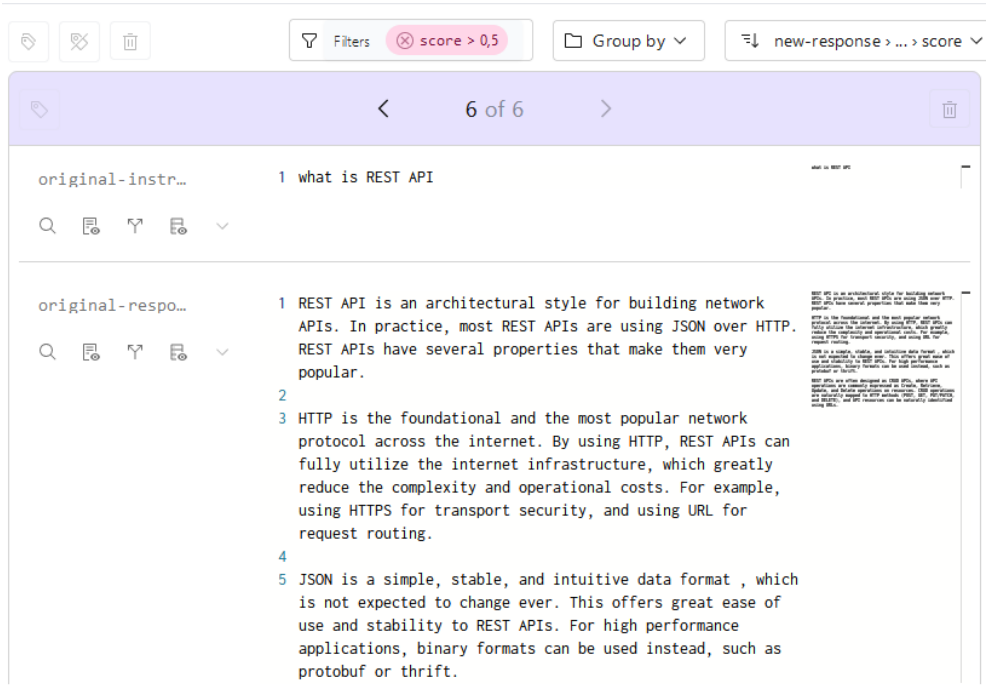


Figure 3.3: Flow chart representing the annotation process. The color coding represents the different types of data sources.

The automated annotation techniques were done using `Lilac`. This includes, for translation, automatically annotating the language of each example and removing non-English examples. As demonstrated in Figure 3.4, detecting code and translation examples was effectively done by semantic keyword searches to detect translation tasks, and using concepts to detect code. Other annotations were performed similarly.

Grammar and spell checks were mostly included as a part of the manual review. However, efforts were made to automatically validate translated examples, as the spelling and grammar of such examples were considerable weaknesses. This was done utilizing an open-source Python library called `spylls` [28], although this did not prove to be highly effective.



(a) Identified questions related to code, with a score of above 0.5, in the Dolly-15k dataset.

(b) Semantic search of ”translation” in the Capybara dataset.

Figure 3.4: Examples of searches with Lilac that allow automatic annotation.

3.5 Data Visualization

Visualizing the collected and cleaned data allows for a general overview of the tasks and subjects involved, as well as establishing the overall quality of the data. This is particularly challenging when working with large text

datasets consisting of several thousand examples. This section presents two main methods established for effective data visualization.

3.5.1 Clusters

Clusters is a feature that can be utilized with `Lilac`. By utilizing a LLM, every example can be clustered into different categories and sub-categories. This helps provide an overview of the topics and tasks that are covered within the data. For instance, Figure 3.5 depicts two categories identified in one dataset. The leftmost side shows the categories identified and the proportion each occupies in relation to the entire dataset. Within the categories, sub-categories are identified, that specifically outline topics and tasks within the larger category. Although clustering provides valuable insights and can be considered another way of annotating data, it was not used in the data annotation process. Instead, it served as a tool for understanding the dataset's structure and content distribution.

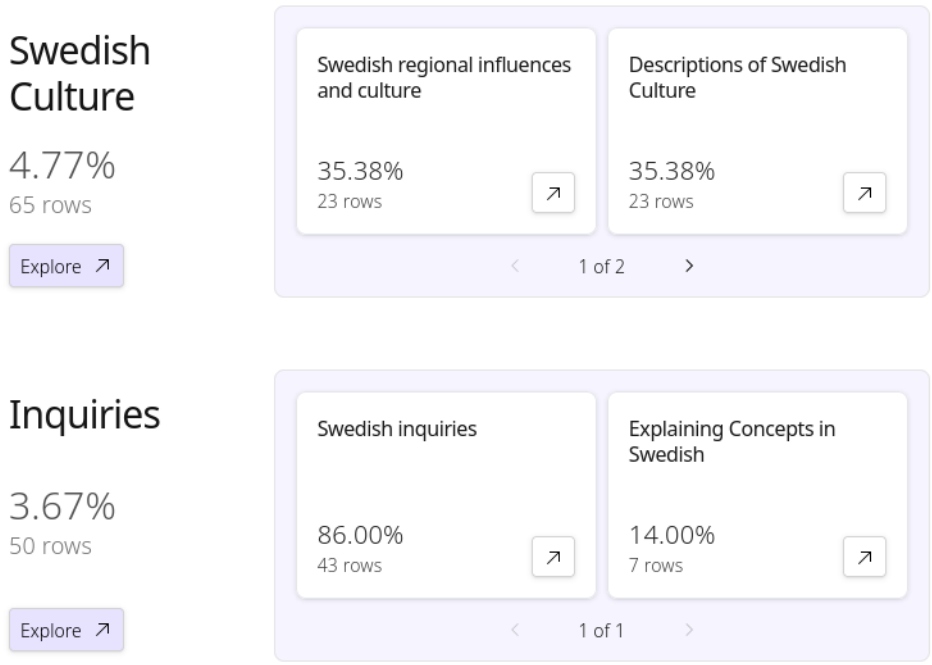


Figure 3.5: Example of two categories identified with Lilac Clusters.

3.5.2 Bunkatopics

Another tool used that utilizes embedding models to visualize data is an open-source tool called `BunkaTopics` [29]. This tool provides an overview of

the topics that are covered in the dataset. Since it is visualized using an embedding model, we can observe clusters with topics that are considered close to each other. For example, Figure 3.6 depicts a visualization of a synthetically generated dataset. To the right, we can observe topics such as words, sentences, and grammar being closely related to Swedish and language. And in turn, Swedish and language are closely related to dialects and folk music.

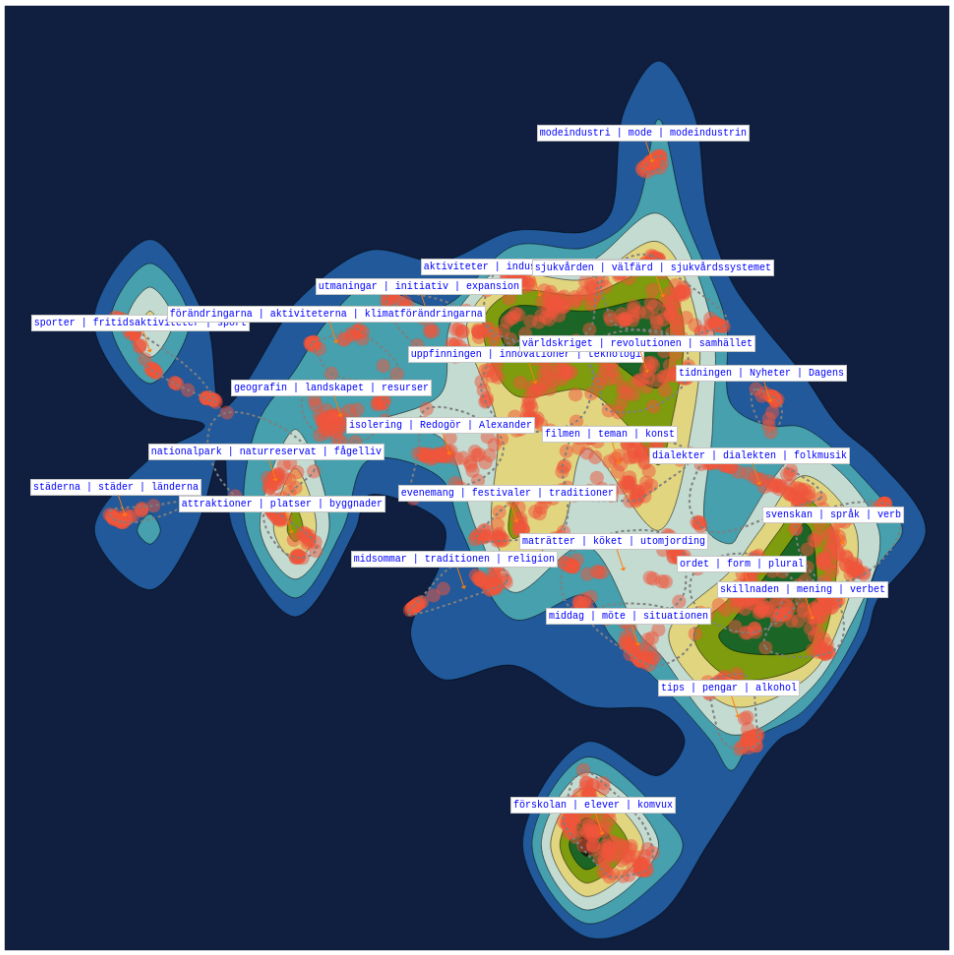


Figure 3.6: A synthetically generated dataset, visualized using BunkaTopics.

Depending on the size of the dataset, we can effectively adjust the amount of clusters and the topics identified within them. The different coloring helps depict the density of each cluster, and hovering over the visualization will present some examples from the corresponding clusters. Using this

visualization, we have the flexibility to construct a new dataset by excluding topics that do not align with the data scope.

3.6 Fine-tuning Setup

This section covers the technical aspects of the fine-tuning environment, including the selection of the pre-trained model, chat template, and configuration of training parameters.

3.6.1 Pre-trained Model Selection

For the pre-trained model, GPT-SW3 from AI Sweden was chosen due to its extensive training on Nordic data [6]. Even though it is based on the older GPT-2 architecture, its strong focus on Nordic linguistic nuances made it preferable over other open-source pre-trained models, as they are typically trained on predominantly American data. For all of the experiments, the 6.7 billion parameter model was chosen as this was considered a balanced trade-off between computational efficiency and performance.

3.6.2 Chat Template

The chat template used for fine-tuning is similar to the template used in already existing instruction models from AI Sweden. Figure 3.7 demonstrates an example with an applied chat template, ready for tokenization. Here, `<s>` represents the beginning-of-sentence token (BOS), and `<|endoftext|>` represents the end-of-sentence token (EOS). These are both defined within the GPT-SW3 tokenizer. It is worth noting that the `EOS` token appears in the beginning of the example, this is because the resulting tensors are packed together during training to optimize efficiency.

Some of the fine-tuning experiments incorporated a system prompt, while others did not. In the cases they did not, the system prompt (if existent) was included as a part of the user prompt.

```
<|endoftext|><s>
SYSTEM:
You are a helpful assistant whose name is Leif.
<s>
USER:
Hello! What's your name?
<s>
ASSISTANT:
Hello! I am Leif, how may I help you?
<s>
```

Figure 3.7: Example of applied Chat Template with BOS-token and EOS-token.

3.6.3 Training Configuration

Both Instruction Tuning and DPO alignment were performed with QLoRA, with either 4-bit or 8-bit quantization and differing LoRA parameters. With both setups, we utilize AdamW Optimizer and Cosine Annealing Scheduler, with an initial learning rate within the range of 10^{-6} to 10^{-4}. The fine-tuning environment was set up using PyTorch, along with the Huggingface Transformers library [30, 31]. The Huggingface PEFT library was used for LoRA setup [32]. All of the fine-tunes were performed using a NVIDIA A100-SXM4-40GB GPU.

3.6.3.1 Instruction Tuning

The general approach taken to instruction tuning was to fine-tune for many amount of epochs and implement early stopping when no further decrease in evaluation loss could be observed.

Below is a list of different hyperparameters used. Considering that these numbers differed between experiments, the hyperparameters are given in spans.

- **Batch Size**: 4-8
- **Gradient Accumulation Steps**: 10-20
- **Effective Batch Size**: 30-160
- **LoRA Rank**: 64, 128, 256
- **Epochs**: 3-20

- **Validation Split:** 20%

The amount of gradient accumulation steps was chosen depending on the desired effective batch size. As a rule, this was kept relatively high to mimic the batch size used for the already existing GPT-SW3 instruction models. With increased gradient accumulation steps, the convergence of the evaluation loss took longer. Training typically took 1 to 4 days.

3.6.3.2 Direct Preference Optimization

For DPO-alignment, the same setup was used for every fine-tuning. This is due to the limited amount of DPO experiments performed, as opposed to Instruction Tuning. Training typically took around 8 hours.

- **Batch Size**: 3
- **Gradient Accumulation Steps**: 20
- **Effective Batch Size**: 60
- **LoRA Rank**: 256
- **Epochs**: 1
- **Validation Split:** 15%

3.7 Evaluation

In this study, three main methods of evaluation were employed. All of them involve measuring the model outputs based on a task from a given dataset. With the Both few-shot and zero-shot evaluations were made.

3.7.1 ScandEval

Seven different evaluation tasks in Swedish were performed using the ScandEval benchmark suite [22]. These include three original Swedish datasets and four translated datasets. ScandEval does not utilize a chat template but instead relies on the model's few-shot capabilities. Below is a list detailing the different datasets along with what they evaluate, how they are used for evaluation, as well as the main metric used.

- **SUC3 - Swedish Named Entity Recognition**

  Task: Identify and categorize entities in text, such as names of people, organizations, locations, etc. This is used to evaluate the model's ability to recognize entities in text, demonstrating how well it can understand and extract relevant information from pieces of text.

  Metric Used: Micro-Averaged F1-Score

- **SweReC - Swedish Sentiment Classification**

  Task: Classify the sentiment of a given example as either positive, negative, or neutral. This is used to evaluate the model's ability to identify emotional tone in Swedish text.

  Metric Used: Matthew's Correlation Coefficient (MCC)

- **Scala-SV - Swedish Linguistic Acceptability**

  Task: Given a sentence, determine whether it is grammatically correct or not.

  Metric Used: Matthew's Correlation Coefficient (MCC)

- **ScandiQA-Sv - Question Answering**

  Task: Given a small piece of text, answer a question about it in a trivia-like manner.

  Metric Used: Exact Match (Accuracy)

- **SweDN - Swedish Summarization**

  Task: Summarize the given examples. This is used to measure the model's ability to condense Swedish text while maintaining the essential parts of the text. The evaluation is done using BERTScore, which evaluates the semantic similarity of the summarized text to a corresponding human-written one.

  Metric Used: BERTScore

- **MMLU-Sv - Knowledge**

  Task: Answer questions correctly, similar to ScandiQA, however with four alternatives presented to the model and no prior information given.

  Metric Used: Matthew's Correlation Coefficient (MCC)

- **HellaSwag-Sv - Commonsense Reasoning**

  Task: Given an incomplete example, choose the correct completion between four given options.

  Metric Used: Matthew's Correlation Coefficient (MCC)

The benchmark runs the evaluation on each dataset 10 times using different evaluation metrics depending on the task. The score is presented as a mean value of the 10 runs, along with a confidence interval.

3.7.2 BiaSWE

BiaSWE is a dataset presented by AI Sweden consisting of 450 examples in Swedish annotated with misogyny or hate speech [33]. Using this dataset, the model was asked to classify each example with either hate speech, misogyny, both, or neither. The evaluations were made using few-shot prompts and measured using F1-score averaged over 10 runs. Figure 3.8 illustrates how prompting was done, however with the few-shot examples oppressed due to its harmful nature.

Identifiera följande mening som hets mot folkgrupp och/eller misogyni. Svara enligt:
hets mot folkgrupp: Ja/Nej
misogyni: Ja/Nej
`<few-shot examples>`

Figure 3.8: Prompt demonstration for BiaSWE evaluation.

3.7.3 SweSAT (Högskoleprovet)

The Swedish Scholastic Aptitude Test (SweSAT) is a standardized test in Sweden used for university admissions. It consists of a verbal part, testing reading comprehension and word knowledge, and a quantitative part testing mathematics and logical thinking.

Three different categories from the verbal parts were extracted from 13 SweSAT tests between fall 2013 and spring 2019. These include the three categories *ORD*, *MEK*, and *LÄS*. These three were specifically chosen with the motivation that they explicitly measure Swedish language capability. They test the following:

- Word Comprehension (ORD): Given a word or expression, pick the option that best corresponds to its meaning. Choose between 5 options.
- Reading Comprehension (LÄS): Given a long text, answer questions about the content of the text. Choose between 4 options.
- Sentence Completion (MEK): Given a text with gaps, fill in the words that should be in the gaps. Choose between 4 options.

The models were evaluated with both few-shot and zero-shot prompts. However, due to the limited context window of 2048 tokens, the *LÄS* part was only evaluated within a zero-shot setting due to the long text being part of the prompt. The performance was evaluated over 10 runs, using accuracy as the primary measurement. Since this evaluation setup only includes a subset of the verbal part, using a standardized score as part of the evaluation was not considered.

Chapter 4

Results and Analysis

This chapter presents the results acquired in this study. First, an overview of the data collected and the experiments conducted are presented. After that, the results from the performed evaluations and benchmarks are presented.

4.1 Dataset Summary

The datasets used in this study can be categorized into three types: translated, original Swedish, and synthetic. Table 4.1 provides a summary of these datasets. A total of five translated datasets (one of them consisting of DPO-pairs), two original datasets, and three synthetic datasets were collected. The size of the data refers to the final size since some datasets have been iterated upon with added and removed examples. All of the translated datasets are derived from existing open-source datasets, ending with ”SV” to denote their translation to Swedish.

Full details of each dataset, including their origins, content specifics, and curation methods, can be found in Appendix A.2.

Type	Dataset Name	Brief Description
Translated	SlimOrca-SV	Curated instruction data with system prompts (∼35k examples)
Translated	CamelAI-SV	Domain-specific instruction data in science and math (∼7k examples)
Translated	Pure-Dove-SV	Multi-turn conversations (∼2.8k examples)
Translated	OpenHermes-SV	Diverse instruction dataset collection (∼28k examples)
Translated	Orca-DPO-Pairs-SV	DPO-pairs derived from OpenOrca (∼8k examples)
Original	BibblanSvarar	Q&A from Swedish libraries (∼4.1k examples)
Original	HP-ORD	Swedish SAT vocabulary questions (∼3k examples)
Synthetic	BibblanSvarar-Synthetic	Generated based on BibblanSvarar questions (∼4.1k examples)
Synthetic	swedish-instruct-data-chatgpt4	Sweden-related Q&A pairs (∼1.3k examples)
Synthetic	BezzerWizzer	Swedish board game Q&A (∼0.5k examples)

Table 4.1: Overview of datasets used in the study

4.1.1 Dataset Combinations for Training

Table 4.2 shows an overview of the datasets that were created, as well as the sources they contain. Some of the datasets have been iterated upon, while others have been completely discarded after experimentation. The ones that were ultimately discarded were so either due to lacking translation quality or bad overall quality. Specifically, these datasets include the first *OpenHermes + HP* dataset, as well as *BibblanSvarar*. However, the latter was later reworked to incorporate synthetic answers. Prior to fine-tuning, these datasets have been added together in different combinations to make up a larger dataset. To keep track of these larger datasets, they were given the name *"hopkok"*, however with the exception of the first dataset which was regarded as an initial test experiment.

Dataset Name	Dataset Sources	Size
OpenHermes + HP	OpenHermes-SV HP-ORD	∼28k
hopkok-v1	SlimOrca-SV-33K Pure-Dove-SV Swedish-instruct-data-chatgpt4 BibblanSvarar	∼41k
hopkok-v2	SlimOrca-SV-35K Pure-Dove-SV Swedish-instruct-data-chatgpt4 CamelAI-SV-7k BezzerWizzer-0.25k	∼45k
hopkok-v3	SlimOrca-SV-35K Pure-Dove-SV Swedish-instruct-data-chatgpt4 CamelAI-SV-7k BezzerWizzer-0.5k BibblanSvarar-Synthetic	∼50k

Table 4.2: The main datasets used for training.

4.2 Experiment Summary

A total of five major instruction fine-tunes were performed from the pre-trained GPT-SW3-6.7B model. Out of these, two were further fine-tuned with DPO alignment.

4.2.1 Instruction Tuning Experiments

Table 4.3 presents an overview of the fine-tunings along with relevant parameters.

In the initial experiments, the effective batch size was kept relatively high to ensure sufficient data per iteration. However, this was decreased throughout the iterations, to observe impact on performance, and then increased again for the final experiments. The same idea was applied to the number of epochs, typically because increased gradient accumulation steps result in a slower reduction of evaluation loss.

Additionally, a system prompt was included in the chat template for two of the experiments. However, after observing that the inclusion of a system

prompt did not lead to a substantial increase in evaluation performance, it was discarded for the final two experiments.

For the LoRA parameters, 4-bit quantization was kept initially, with the `nf4` data type. However, this was later changed to 8-bit quantization. The rank of the LoRA matrices was kept at 128 for a majority of the experiments to ensure a sufficient trade-off between batch size and the number of trainable parameters.

Exp #	Dataset	Sys. Prompt	Batch Size	Loss	Epochs	LoRA Rank	QLoRA Quant.
1	OH + HP	No	160	1.358	20	64	4-bit
2	hopkok-v1	Yes	120	1.408	12	128	4-bit
3	hopkok-v2	Yes	30	1.290	3	128	8-bit
4	hopkok-v2	No	60	**1.285**	6	128	8-bit
5	hopkok-v3	No	100	1.299	6	128	8-bit

Table 4.3: Overview of the instruction tuning experiments performed. The loss refers to the final evaluation loss. Batch Size refers to gradient accumulation steps + device batch size.

4.2.2 DPO Experiments

Table 4.4 depicts information about the two main DPO alignments performed. The training was done for one epoch, which was suggested to be sufficient by the authors of the DPO paper [17]. This proved efficient, as the evaluation accuracies (meaning the ratio of which the preferred and rejected answer was identified) were fairly high. The batch size was kept at 60, which was deemed sufficient due to the low amount of training data. None of the DPO fine-tunes, nor their base models included system prompts as part of the applied chat template.

Exp #	Dataset	Base Model	Batch Size	Acc.	Epochs	LoRA Rank	QLoRA Quant.
1	Orca-DPO-SV	hopkok-v2	60	0.98	1	256	8-bit
2	Orca-DPO-SV	hopkok-v3	60	**0.99**	1	256	8-bit

Table 4.4: Overview of the two DPO experiments performed. Acc. refers to the final evaluation accuracies, meaning the model's ability to identify the preferred response over the rejected one during evaluation. Both experiments, as well as the base models, did not utilize system prompts.

4.3 Evaluation

4.3.1 ScandEval

In this subsection, results from the ScandEval evaluations are presented along with key observations. The following figures depict bar charts for different evaluation datasets from the ScandEval benchmark suite. All of the model metrics have been averaged over 10 runs, and all of the evaluations were done using few-shot prompts, with **no** chat templates applied. For simplicity, models produced within this project are referred to as *experiment models*. The GPT-SW3 base model, along with its official instruct model, are referred to as *baseline models*. The scores are represented by different metrics, however, all of the scales of the scores range from 0-100, where 100 represents the best possible score.

Text Classification

In Figure 4.1, the results from the sentiment classification dataset **SweReC** are presented. The models were evaluated using MCC, where the -1 to 1 scale has been normalized to fit a 0-100 scale. This means that a score of 50 represents a random guess. In this task, the model was prompted to identify the sentiment in a given text. Below is an example of one of the individual shots used in a few-shot prompt for this evaluation:

```
Recension: Lika bra varje gång
Sentiment: positiv
```

The best-performing model is the GPT-SW3 base model (77.47). However, all of the experiment models outperform the baseline GPT-SW3 instruct model,

with the best experiment model being the *hopkok-v3* model (75.97). This translates roughly to 0.5 on the regular MCC scale, indicating a moderately positive correlation. It is also notable that both DPO fine-tunes resulted in worse performance than the rest of the experiment models.

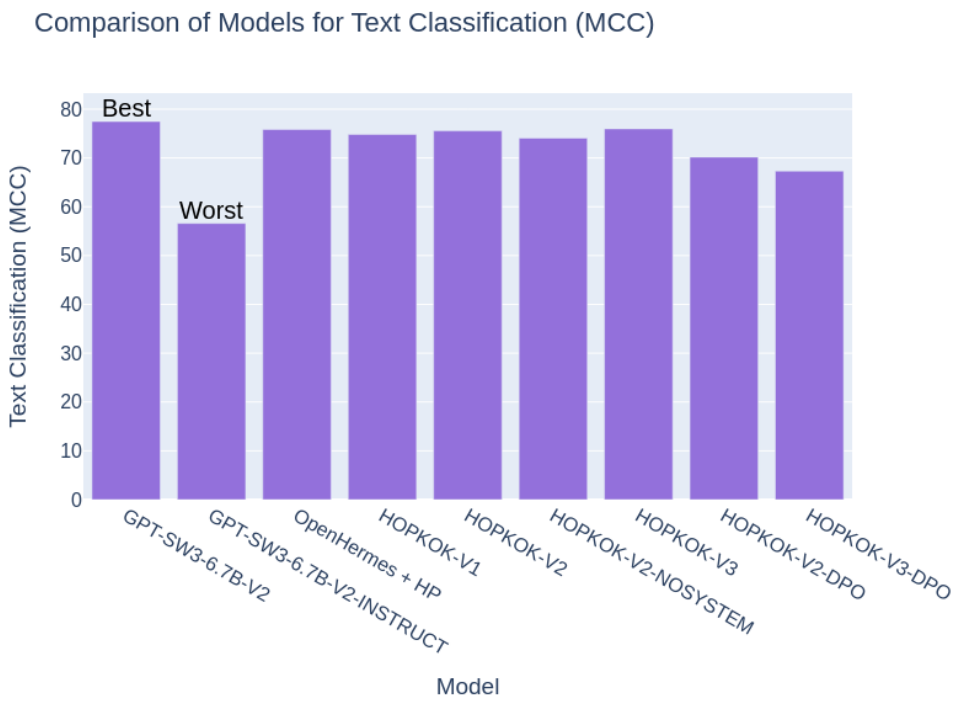


Figure 4.1: Comparison between the baseline models and the experiment models on text classification tasks (few-shot). The evaluation was done by calculating the Matthews Correlation Coefficient, normalized to fit the 0-100 scale.

Information Extraction

Results from the Named Entity Recognition dataset **SUC3** are presented in Figure 4.2. The model was prompted to identify named entities and classify them as e.g. person, organization, or location. This can be exemplified in the following prompt:

```
Mening: Byar i lidläge vid de stora skogssjöarna
Bygdeträsket och Göksjön.
Namngivna entiteter: {"person":[],
"plats":["Bygdeträsket, Göksjön],
```

```
"organisation":[],
"diverse":[]}
```

Here, *hopkok-v3*, along with some other experiments, outperforms the baseline models. Again, the baseline instruct model demonstrates subpar performance in comparison to the other models.

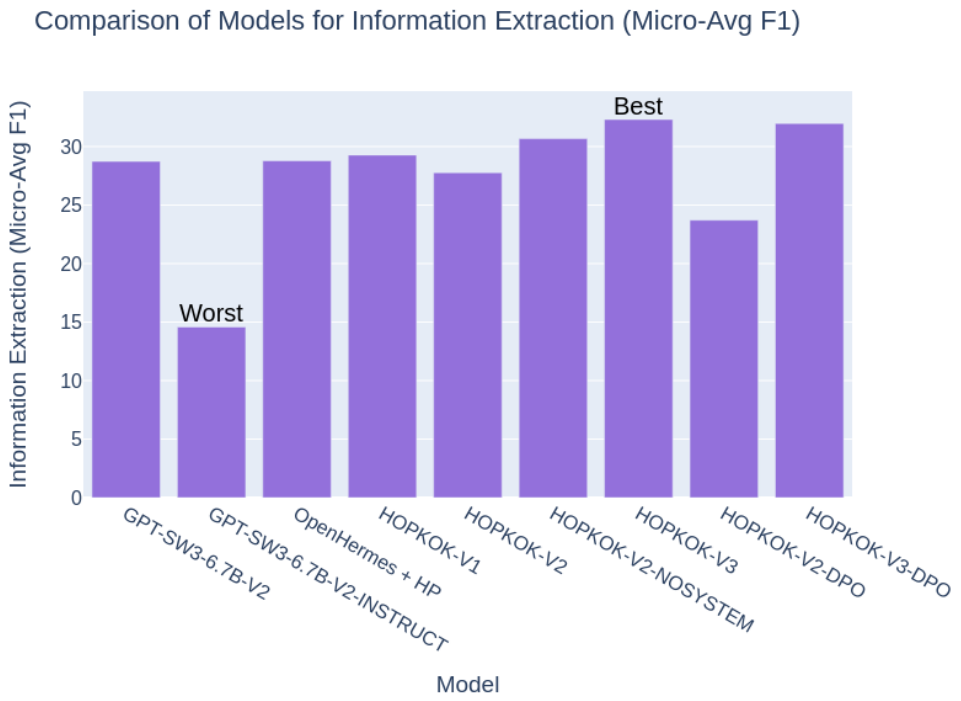


Figure 4.2: Comparison between the baseline models and the experiment models on information extraction tasks (few-shot). The evaluation was done by calculating the Micro Averaged F1-score.

Grammar

The results from the linguistic acceptability dataset **Scala-SV** are presented in Figure 4.3. In this task, the model was tasked with classifying a given sentence as either grammatically correct or incorrect, as exemplified below. This was measured using MCC, normalized to fit the 0-100 score.

```
Mening: Var får man tag i dem i lilla Sverige?
Grammatiskt korrekt: ja
```

Overall, the performances on this task were significantly poor, with scores ranging from 1-10, indicating almost inverse correlation. The evaluations show that the baseline instruct model performs best, with only *hopkok-v1* and *hopkok-v2-nosystem* outperforming the pre-trained base model by a small margin.

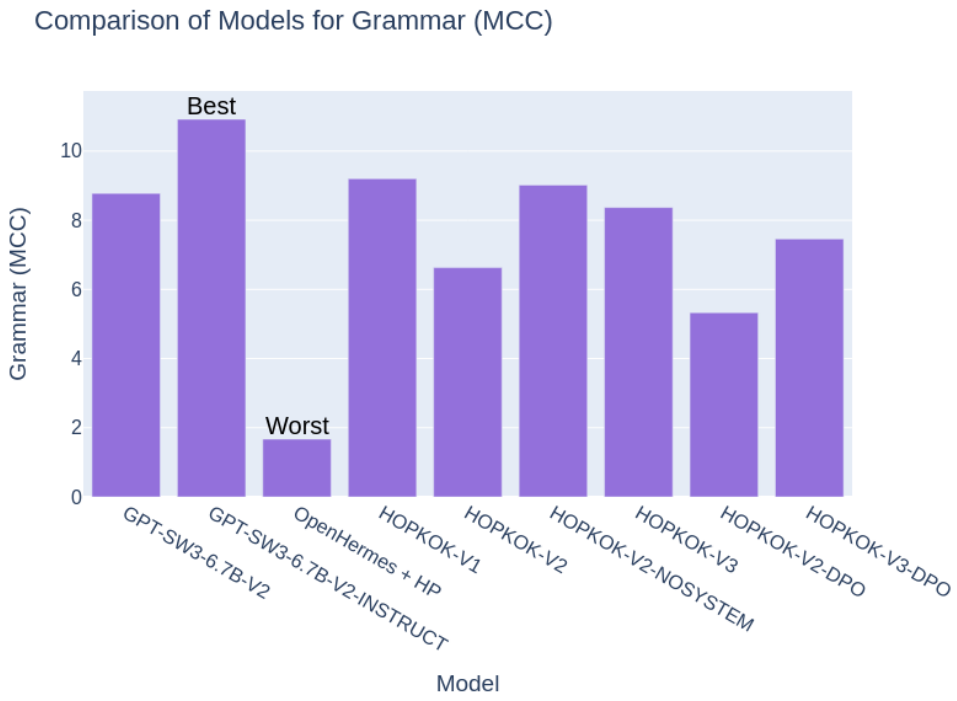


Figure 4.3: Comparison between the baseline models and the experiment models on grammar tasks (few-shot). The evaluation was done by calculating the Matthews Correlation Coefficient, normalized to fit the 0-100 scale.

Question Answering

The results from the question-answering dataset **ScandiQA-SV** are depicted in Figure 4.4. The model was tasked with answering trivia questions based on a small piece of information, without being presented with alternatives. If the exact answer was found within the generated output, it was considered a match. This was measured by calculating the proportion of exact matches. Below is an example of a few-shot prompt:

```
Text: "(Sittin' On) The Dock of the Bay" är
en låt som är skriven av soulsångaren
```

```
Otis Redding och gitarristen Steve Cropper.
Fråga: Vem sjöng Sitting on the dock of the bay?
Svar på max 3 ord: Otis Redding
```

The evaluations show somewhat consistent results, with some of the experiments outperforming the baseline models by a small margin. Most models answer the questions correctly around half the time, indicating a moderate level of knowledge.

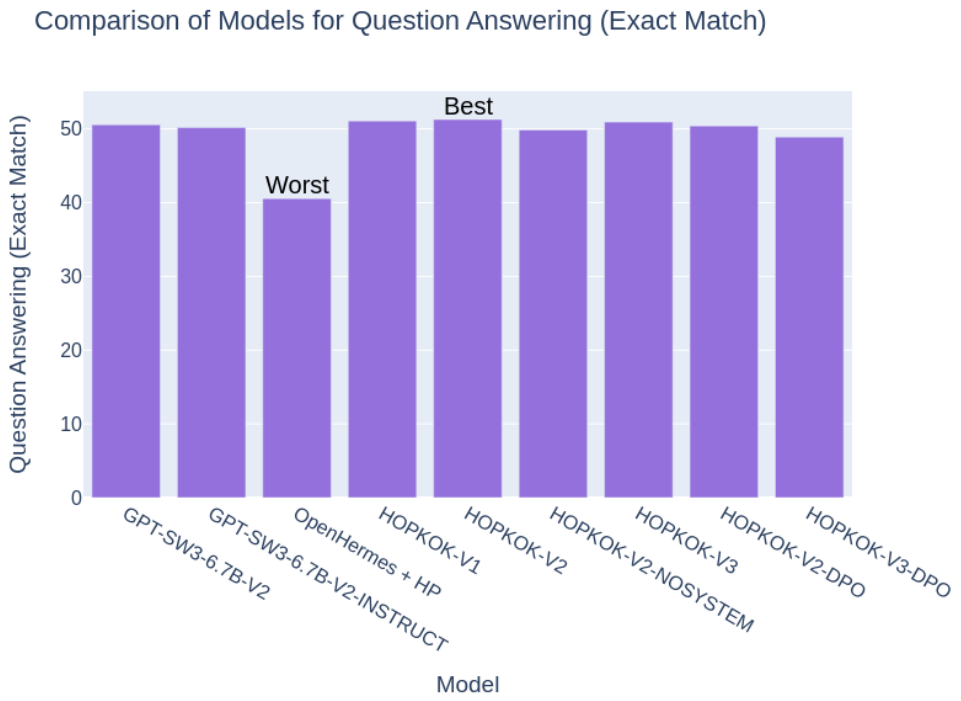


Figure 4.4: Comparison between the baseline models and the experiment models on question answering tasks (few-shot). The evaluation was done by calculating the proportion of exact matches.

Summarization

The results from the Swedish summarization dataset **SweDN** are shown in Figure 4.5. The model was tasked with summarizing a given text, and the BERTScore was calculated by comparing the semantic similarity of the summary to that of a human-written counterpart using contextual embeddings. To perform the calculation, ScandEval utilizes an external BERT-model, abstracted from the user, on the generated GPT-output.

The evaluations show overall that the baseline models outperform all of the experiment models. However, both DPO-finetunes have resulted in boosted performance. The best-performing models have a BERTScore of around 60, indicating a moderate level of similarity to the reference summary.

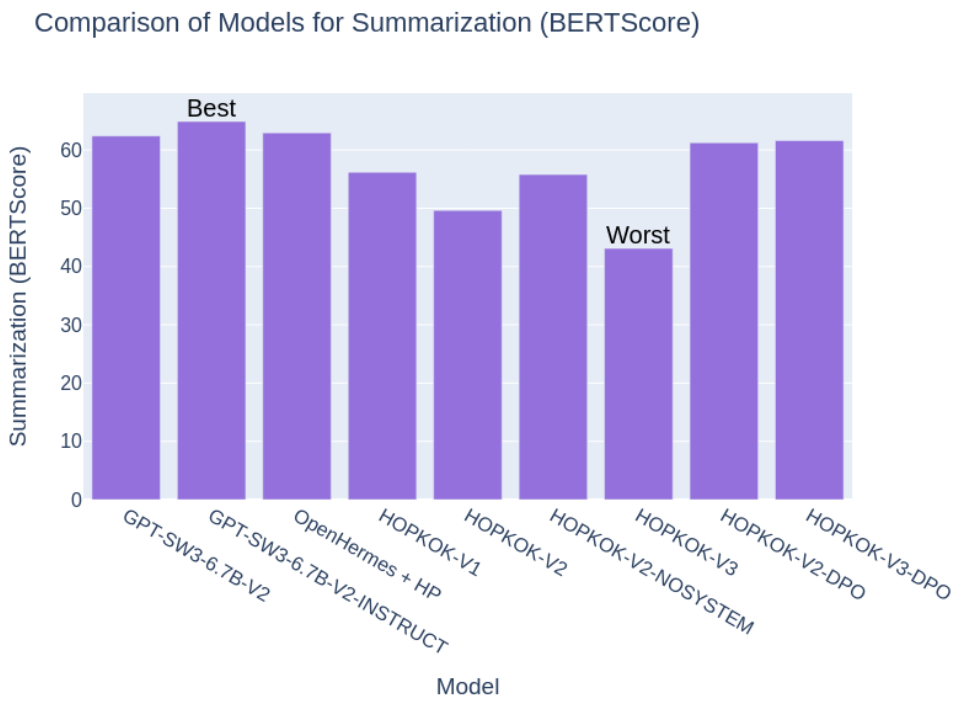


Figure 4.5: Comparison between the baseline models and the experiment models on summarization tasks (few-shot). The evaluation was done by calculating the BERTScore.

Knowledge

In the knowledge dataset **MMLU-SV**, shown in Figure 4.6, the model was tasked with answering questions given four alternatives, and evaluated with MCC. Below is an example of one of the individual shots used in a few-shot prompt:

```
Fråga: Termen babyboomgenerationen hänvisar
till personer som
Svarsalternativ:
a. Föddes före andra världskriget
b. Har haft en extraordinär mängd avkomma
```

```
c. Föddes precis efter andra världskriget
d. Har uppfostrat sina barn i små byar
Svar: c
```

A trend of increased performance can be observed throughout the experiment iterations. The *hopkok-v3-dpo* model has also increased its performance compared to its *hopkok-v3* base model. However, like the grammar evaluations, the MCC scores show an almost negative correlation; indicating no better than a random guess. This was also confirmed when later observing accuracies, which showed around 25%, with *hopkok-v3-dpo* having roughly 32% accuracy.

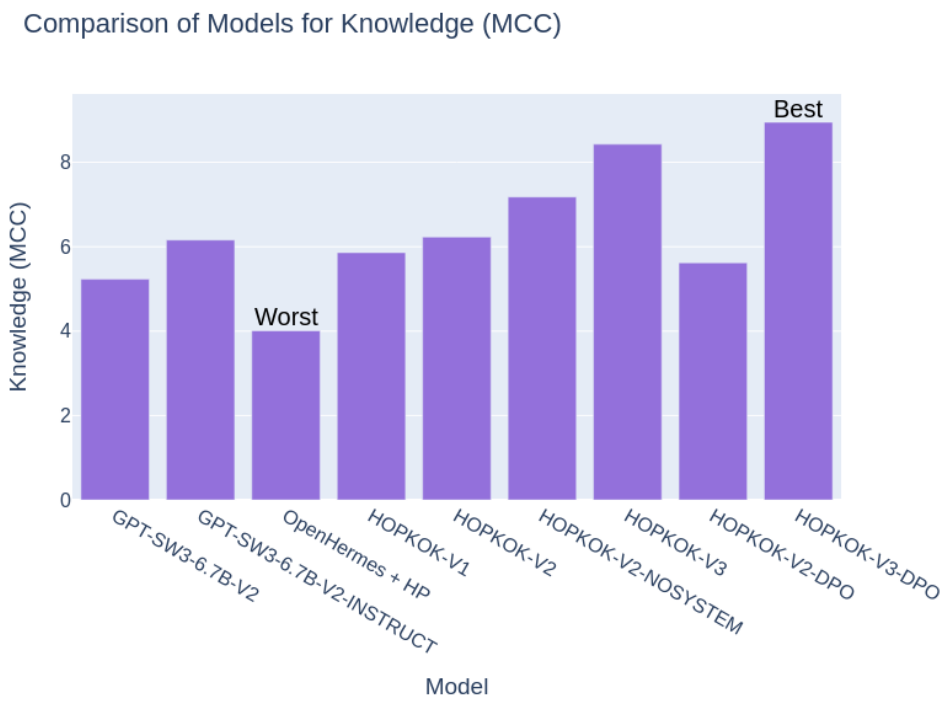


Figure 4.6: Comparison between the baseline models and the experiment models on knowledge tasks (few-shot). The evaluation was done by calculating the Matthews Correlation Coefficient, normalized to fit the 0-100 scale.

Reasoning

Results from the **HellaSwag-SV** dataset are shown in Figure 4.7. The answers were evaluated using MCC, again normalized to fit the 0-100 scale. Below is a demonstration of one of the shots used in the few-shot prompts.

```
Fråga: En man krattar löv i en trädgård.
Han använder en transparent påse för att
packa ihop dem. han
Svarsalternativ:
a. använder en stor hink för att tvätta löven.
b. knäböjer och lägger det han krattar på marken.
c. flyttar sedan påsarna från trädgården och
klipper gräsmattan.
d. skrapar ihop löven och lindar dem med tejp.
Svar: c
```

The evaluation shows an increase in model performance across all experiments compared to both baseline models. Both DPO-finetunes also demonstrate increased performance compared to their base models. While *hopkok-v3-dpo* shows the most substantial increase in performance, it reaches a score of 33. This roughly corresponds to -0.7 on the regular MCC scale, indicating a negative correlation.

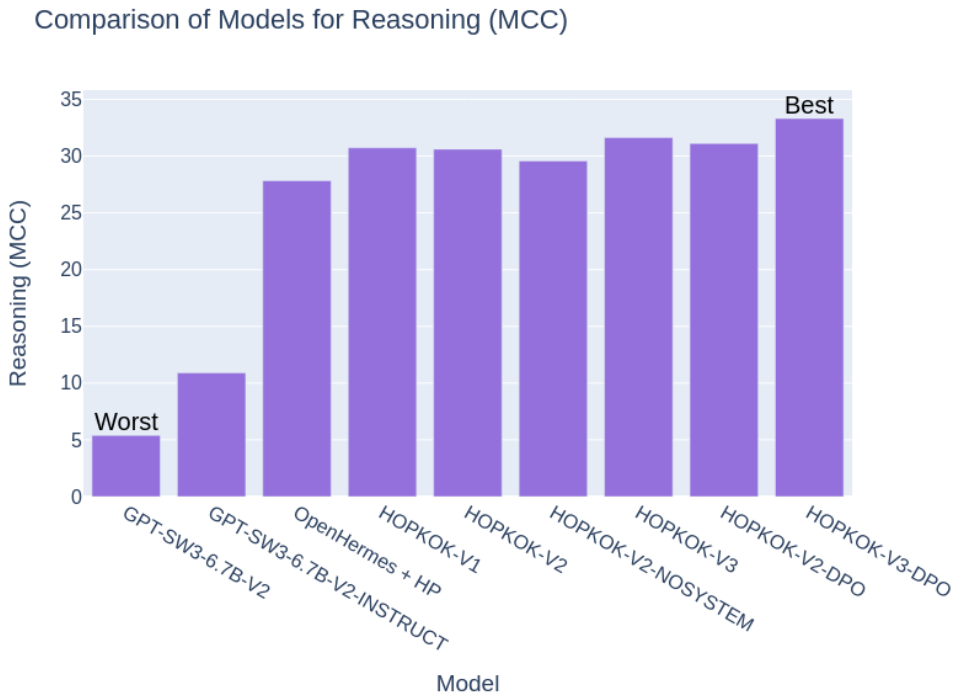


Figure 4.7: Comparison between the baseline models and the experiment models on reasoning tasks (few-shot). The evaluation was done by calculating the Matthews Correlation Coefficient, normalized to fit the 0-100 scale.

4.3.2 BiaSWE

The BiaSWE evaluations were performed using only the GPT-SW3 Instruct model as the baseline, as all of the evaluations were done using an applied chat template along with 3-shot prompts.

Figure 4.8 demonstrates the F1-scores, averaged over 10 runs, on hate speech and misogyny respectively across all the models. All of the experiment models were better at identifying hate speech than misogyny. Both DPO-finetunes became better compared to their base models, with *hopkok-v3-dpo* being the best model overall at identifying both hate speech and misogyny. While it is apparent that the baseline instruct model performs the worst, it identifies misogyny more often than hate speech.

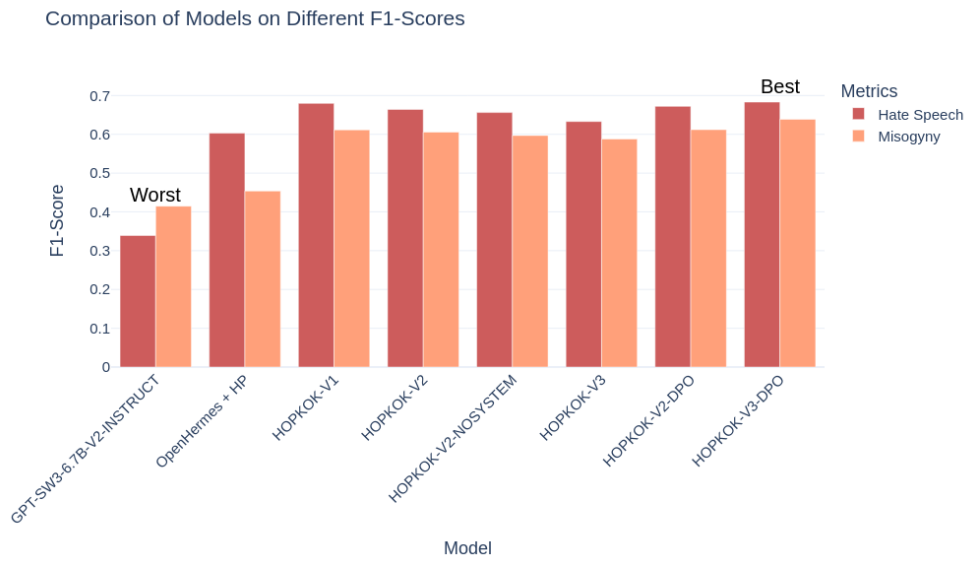


Figure 4.8: Comparison of the model's capabilities of identifying Hate Speech and Misogyny (F1-score). Best and worst models are indicated for both hate speech and misogyny.

4.3.3 SweSAT

SweSAT evaluations were performed using only the GPT-SW3 Instruct model as the baseline. Due to the inclusion of a chat template, the pre-trained GPT-SW3 model was not included. Both five-shot and zero-shot evaluations were made. However, due to the limited context window, the *LÄS* parts were only evaluated with zero-shot prompts. The metric used was accuracy, with a scale ranging from 0 to 1.

4.3.3.1 Word Comprehension (ORD)

Figure 4.9 illustrates the results from both zero-shot and five-shot evaluations on the word comprehension part. The model was given a word and asked to pick the meaning of the word among five alternatives. Overall, the models perform better in a zero-shot setting, with *hopkok-v3-dpo* demonstrating the best zero-shot performance along with the lowest variance. However, many models are close to random selection (0.2), especially in a five-shot setting.

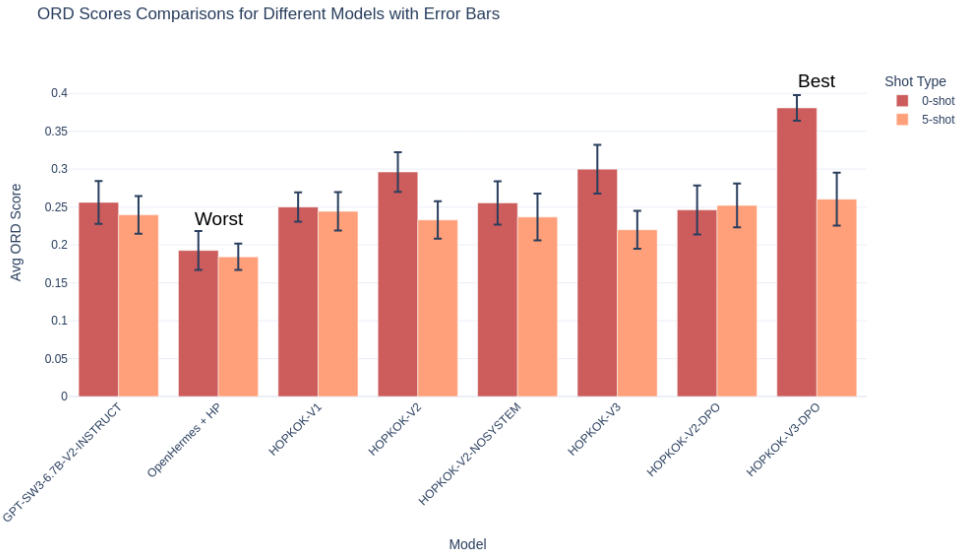


Figure 4.9: Comparison of model performance on Word Comprehension, showing accuracy scores (scale 0-1) along with standard deviations outlined as error bars. Best and worst model is indicated for both 0-shot and 5-shot evaluation.

4.3.3.2 Reading Comprehension (LÄS)

Figure 4.10 illustrates the results from five-shot evaluations on the reading comprehension part. The model was given a long text and was then asked questions about the text, along with four options. An improving trend can be seen throughout every model iteration, although with only the *hopkok-v3-dpo* model outperforming the baseline instruct model by a small margin.

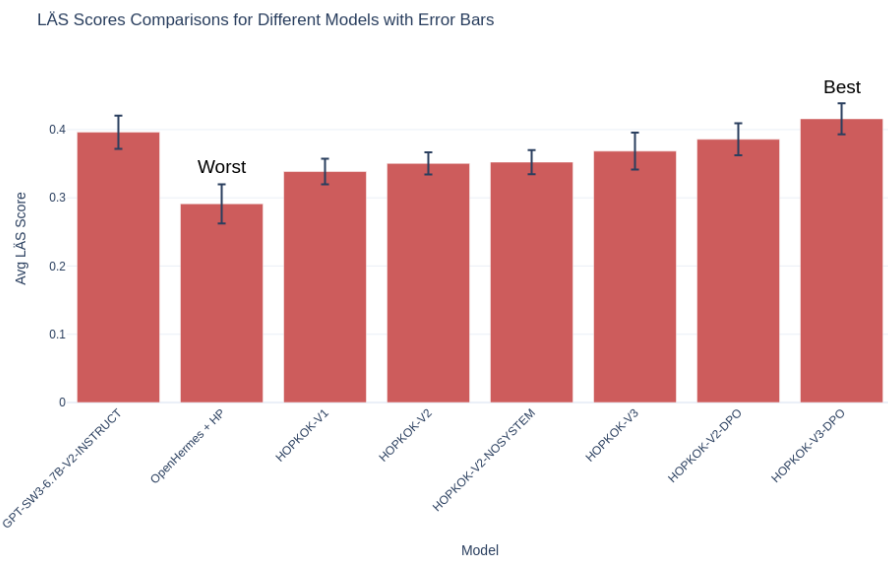


Figure 4.10: Comparison of model performance on Reading Comprehension, with 5-shot prompts, showing accuracy scores (scale 0-1) along with standard deviations outlined as error bars.

4.3.3.3 Sentence Completion (MEK)

Figure 4.11 illustrates the results from both zero-shot and five-shot evaluations on the Sentence Completion parts. The model was given a text with blank spaces and was given four possible options that fill the blank spaces.

In contrast to the ORD scores, the different shot types fluctuate with five-shot demonstrating better performance in some cases. Similar to the other scores, the zero-shot performance of the *hopkok-v3-dpo* displays the best performance of the experiment models, although the baseline instruct model outperforms it by a small margin. A lot of models are also close to random guesses (0.25).

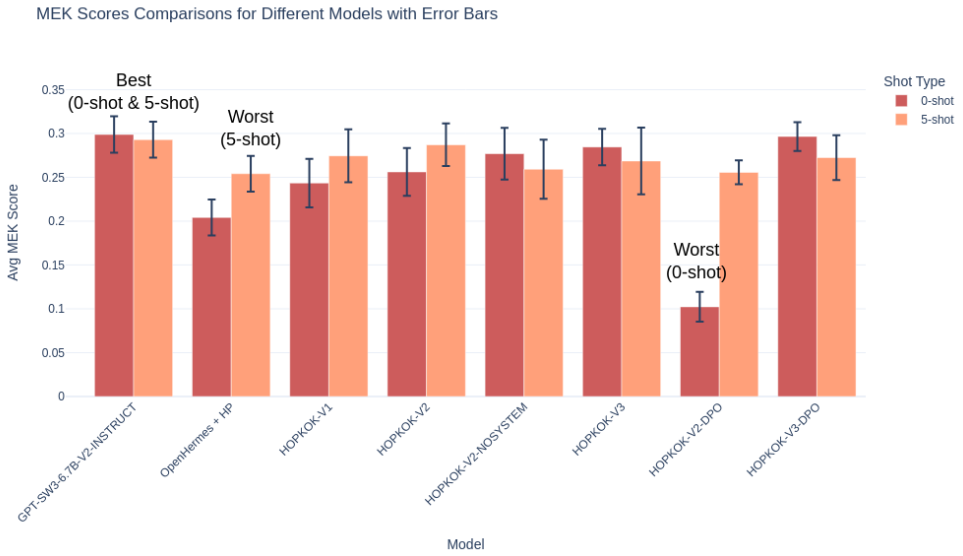


Figure 4.11: Comparison of model performance on Sentence Completion, showing accuracy scores (scale 0-1) along with standard deviations outlined as error bars.

Chapter 5

Discussion

This chapter provides an interpretation of the results, discussing key implications and significant observations.

5.1 Benchmark Summary

The ScandEval evaluations revealed mixed results. Our experiment models outperformed the baseline in four tasks but showed similar or worse performance in three tasks. Specifically, in information extraction (Figure 4.1) and text classification tasks (Figure 4.2), the baseline instruct model *(GPT-SW3-6.7b-v2-instruct)* demonstrates subpar performance compared to both the baseline pre-trained model *(GPT-SW3-6.7b-v2)* and the experiment models. Interestingly, these are two out of three tasks in the ScandEval benchmark with datasets created originally in Swedish (SUC3 & SweReC). In both of these tasks, a majority of the experiment models show better or similar performance to the baseline pre-trained model; despite the lack of included chat template. Moreover, we can see a trend of better performance throughout model iterations, which might suggest a result of better curated and correct Swedish data.

Contradictory to the insights from the aforementioned tasks, the experiment models demonstrated worse performance than both baseline models in Swedish grammar tasks (Figure 4.3). While the dataset used for this task (Scala-SV) is based on data created originally in Swedish, the authors of ScandEval have not been able to verify its grammatical correctness [22]. Therefore, the contradicting results from this task may come from a lack of validity and reliability in the task and its dataset. Regardless, it is noteworthy that the baseline instruct model performs best in this task

while performing significantly worse in the other tasks with datasets created originally in Swedish.

The rest of the tasks in ScandEval, with translated datasets, show either a small increase in performance or not at all. Since these datasets have been translated from English datasets, using DeepL, the quality of these translations can be put into question. For instance, in the summarization task (Figure 4.5), the best-performing model out of the experiment models was the *OpenHermes + HP* model. This model was considered a test model/dataset and has overall shown the worst performance in the rest of the evaluations. One explanation of this outlier result can be that this model is ”closest” to the base model, considering that it had the lowest amount of trainable parameters. Nonetheless, we see a rather large increase in all of the experiment model performances in both knowledge (Figure 4.6) and reasoning (Figure 4.7) tasks; with DPO-finetunes boosting that performance even further. One possible explanation behind this is the way these tasks are presented to the model. Both of these tasks present the model with four options A — D, and the *hopkok-v3-dpo* model performs well in these kinds of tasks. This observation is further emphasized by the SweSAT evaluations (Section 4.3.3), where the zero-shot performance of *hopkok-v3-dpo* exceeds all the other models in the *ORD* part (Figure 4.9) with an average accuracy of 38%, whereas no other model exceeds 30% accuracy, with the baseline having only 25%. The *MEK* part (Figure 4.11), however, shows similar results to the baseline, with both *hopkok-v3-dpo* and the baseline having an average accuracy of roughly 30%. And while the five-shot performance in these tests shows rather unstable results, *hopkok-v3-dpo* still outweighs the baseline instruct model by a small amount (42% compared to 40% accuracy) in the *LÄS* part (Figure 4.10). Overall, the models performed considerably better in a zero-shot setting, which is consistent with the findings from Wei et al [3], where it is suggested that fine-tuned language models perform better in such settings.

Addressing the results from the *BiaSWE* evaluation (Section 4.3.2), the DPO-aligned models demonstrate an increased ability to identify both hate speech and misogyny in the input text. This might suggest that these models are better aligned. While evaluating human alignment is not in essence part of the scope of this thesis, the inclusion of DPO datasets has shown an increased performance in some evaluation tasks. Despite that, a more thorough analysis would be needed to ensure that the model is more aligned e.g. in the sense of producing toxic output.

While the focus of this thesis is on the relative performance of the models,

the performance boosts observed are still considered poor overall performance when put in a wider context. This can be best explained by the outdated GPT-2 architecture and its 6.7 billion parameters. Even though increased performance with low variances has been observed, many evaluations represent random guesses statistically.

In summary, out of all the seven models fine-tunes created, the final instruct model *hopkok-v3* and its DPO-tuned variant *hopkok-v3-dpo* has overall the most stable and best performance across the evaluations performed. While the DPO-tuned variant outperforms the non-DPO-tuned model in a lot of specific evaluation tasks, in the cases it does not, it performs rather badly compared to other experiment models.

5.2 Assessing the Pipeline

5.2.1 Instruction Data

Addressing **RQ1**, where we asked about the control measures required to ensure a reliable Swedish instruction dataset of high quality, we look into the key variables identified that have provided higher-quality datasets throughout the model iterations.

First, the quality of translated datasets plays a significant role in capturing both linguistic and cultural nuances of the Swedish language. The translation model that was utilized produced mostly good translations. Even so, if the datasets are originally from American sources, they carry biases that can impact the quality despite the translations being good. To tackle these biases, we explored synthetic data generation and the use of data originally created in Swedish. The former showed promising results, and the latter was significantly more challenging to gather. Using ChatGPT-4 to generate both Q&A pairs based on given topics gave good examples, however, it also resulted in some grammatically and factually incorrect examples. Gathering data from a Q&A forum such as *BibblanSvarar* resulted in the model outputting more links, as this was apparent in a lot of these examples. Synthetically generating answers based on the questions from *BibblanSvarar* proved to be a good way to tackle this. This potentially allows the model to catch impurities in questions (such as spelling errors) as this was apparent in the questions, while still understanding and answering them desirably. The drawback of this is that it requires a model that produces good answers in Swedish, and while the model used *(Llama-8B-Instruct)* produced a lot of good answers, extensive cleaning had to be

done to remove bad examples. Moreover, generating data from questions of a quiz game such as *BezzerWizzer* did seem to boost model performance in both knowledge and question-answering tasks.

The automatic aspect of the cleaning and curation stage proved efficient and helped produce data of higher quality. This can also be a reason for improved performance throughout iterations, as further cleaning was done to the already existing datasets. Even though an attempt was made to manually annotate examples by inspecting a small subset of the dataset, the overall lack of manual annotation and curation may have led to insufficient improvement in quality. Nevertheless, visualizing the dataset by inspecting the overall topics and tasks helped ensure a general level of quality.

The technical aspect of model training should also be considered. With fine-tuning taking several days, optimizing hyperparameters and achieving a desirable evaluation loss was time-consuming. This trial-and-error approach may have impacted the resulting models.

Finally, the evaluation stage provided an effective strategy for identifying areas that needed improvement and supported the iterative nature of the pipeline. However, evaluation benchmarks such as ScandEval would benefit from further verification, as these datasets also face the aforementioned biases from translated datasets. This is one of the motivations behind formulating the additional evaluations BiaSWE and SweSAT. Despite this, the absence of human evaluation is a significant flaw, as this addition would allow for a more thorough evaluation of cultural and linguistic nuances in model outputs.

In summary, these key variables — improving translation quality, leveraging synthetic data, efficient cleaning and curation, and comprehensive evaluation — enhanced the overall quality of the instruction datasets. However, further refinements would involve incorporating human evaluation and aiming for more refined data collection methods to improve model performance and reliability.

5.2.2 Preference Data

Answering **RQ2**, the inclusion of DPO datasets showed enhanced performance in some tasks, such as answering questions with four given options, as well as an increased capability of identifying harmful text in input sequences. This indicates the potential for integrating preference datasets in the pipeline, however, further analysis is needed to ensure comprehensive human alignment and mitigation of toxic outputs.

The Swedish preference data was gathered solely by translating another

DPO dataset (*Orca-DPO-Pairs*), and no other means of gathering data were explored in terms of preference data. Another alternative to this would be synthetic data generation by letting another model rank model outputs. This was not explored extensively due to the primary focus on instruction data.

Despite these limitations, the extent to which DPO was explored in this study highlights a potential alternative to RLHF for Swedish LLMs. Even with limited resources such as a small preference dataset, and fine-tuning with QLoRA, we managed to see a trend of increased performance, albeit in a few key areas. This performance improvement was unexpected, given that DPO fine-tuning was performed using QLoRA on a model already fine-tuned with QLoRA.

Chapter 6

Conclusions and Future work

6.1 Conclusions

This thesis set out to answer **RQ1:** *What control measures are required to ensure a reliable Swedish instruction dataset of high quality?* by designing a pipeline for creating and curating instruction data in Swedish and evaluating the performance of datasets created from it by fine-tuning LLMs. Further, **RQ2:** *To what extent can preference datasets, such as DPO datasets, be included as part of the pipeline?* was formulated to explore the potential of generalizing the pipeline further with preference datasets.

The findings of this study indicate several key insights into the development and evaluation of Swedish instruction datasets. The quality of translated datasets plays a big role if cultural and linguistic nuances are to be preserved, and biases avoided. Ideally, such datasets should likely be abandoned and focus should be put on tailoring such datasets from anew. Unfortunately, this is likely unpractical without extensive efforts from several instances. Therefore, a translation model that can capture these nuances and source language biases could present a viable alternative. The translation model used in this study has shown potential in this sense, however it lacks thorough evaluation.

The models that have emerged through experimentation demonstrated varying degrees of success across the different evaluations performed. The most notable performance improvements overall were observed in the last iterations with both instruction tuning and DPO alignment, indicating a successful iterative refinement process. Even so, there is a need for better evaluations and benchmarks as existing strategies may lack sufficient validity.

As indicated by Rafailov et al. [17], the use of Direct Preference

Optimization (DPO) shows potential as a resource-efficient alternative to Reinforcement Learning from Human Feedback (RLHF). The DPO-tuned models in this study specifically demonstrated enhanced performance in identifying harmful text and in answering knowledge questions given four alternatives.

Furthermore, the automatic cleaning and curation process, while effective, highlights the need for incorporating human evaluation to capture nuanced linguistic and contextual accuracy that automated methods might miss.

In conclusion, this thesis has demonstrated the potential of a robust pipeline for creating high-quality Swedish instruction datasets and integrating preference datasets to improve LLM performance. While there are areas for improvement, we believe that the methodologies and findings provide a foundation for future work in developing and refining Swedish language models.

6.2 Limitations

It is befitting to shed light on the hardware limitations posed in this study. While a thorough attempt was made to utilize the resources given to its extent, it may have caused a bottleneck in increasing the quality of the models. Techniques like QLoRA have shown to pose a small trade-off in model performance compared to full fine-tuning, but the trade-off is nonetheless existent. This might become problematic when focusing on adjusting the finer details of language models, and it is very apparent that PEFT techniques should be abandoned if any talk is to be made about state-of-the-art models. The attempt to utilize the given resources to its fullest comes with the setback of fine-tuning taking several days, so the time limitation also becomes relevant, and time can be the biggest weakness in this field. Despite this, it is noteworthy that a large part of the results were better than, or as good as, the fully fine-tuned and pre-trained baselines. It is therefore imaginable that the discrepancy in these comparisons could be greater if full fine-tunes were to be performed - which is worth discovering in future work.

Another noteworthy limitation, and general drawback of this study, is the lack of human evaluation. While automatic evaluation is efficient and valuable, it presumably lacks in providing nuanced and subjective analysis of the model outputs the same way that a human would. For example, language experts could offer insights into contextual relevance, tone, and subtle language nuances that automated metrics might miss.

6.3 Ethics

LLMs have the potential to generate textual content of a discriminatory, biased, or untruthful nature. The methods employed in data cleaning and annotation in this thesis have been effective in mitigating such content in the source data. Even so, pre-trained language models have already been exposed to an immense amount of data where such content is difficult to evade. The focus is instead put on employing safety measures and human alignment to make these models as safe as possible. While this thesis has explored DPO as an alternative to the RLHF pipeline, it must be emphasized that all of the models are fully capable of generating untruthful, hurtful, and discriminatory output. Further, the majority of the data used have been gathered from publicly available and open-source origins. However, it is important to note that the original methods and standards used in collecting and curating these open-source datasets are beyond the scope of this thesis and cannot be fully verified.

6.4 Sustainability

When it comes to sustainability, this issue is discussed from three perspectives; namely economic, social, and environmental. The process of training LLMs requires significant computational resources. These issues have been addressed when utilizing parameter-efficient techniques such as QLoRA, which helps reduce the computational load and makes it more energy efficient. Further, this thesis emphasizes the creation of instruction datasets in the Swedish language. This is done partly in an attempt to address social sustainability in terms of linguistic and cultural preservation. By doing so, it also opens up the potential to tackle further issues in these areas, such as the preservation of regional dialects; some of which are at risk of extinction due to a declining number of speakers and which possess unique nuances in vocabulary, grammar, and expression. Lastly, the development of Swedish-specific AI models can benefit businesses or government agencies with solutions tailored to the Swedish language and culture. This can lead to more efficient operations, such as processing government documents. Such initiatives are already set in motion and aim to benefit Swedish society and Sweden's competitiveness within the field.

6.5 Future work

Beyond addressing the limitations discussed in section 6.2, there are several key areas for future work that could enhance the quality and reliability of Swedish instruction datasets and language models.

While technical aspects are crucial, there is a point where the quality of instruction data goes beyond technical considerations and delves into linguistic and cultural aspects. This transition calls for a more extensive effort involving experts from various fields, such as Swedish language experts.

Furthermore, exploring more diverse methods for gathering Swedish data is essential. This could include leveraging crowd-sourcing platforms to collect a broader range of both instruction datasets and preference data and would potentially result in a more diverse and culturally representative dataset.

Lastly, comprehensive benchmarking is another crucial area of improvement. This involves both verifying existing evaluation benchmarks like ScandEval, but also developing new benchmarks that are tailored specifically for the Swedish language.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf [Pages xi, 1, and 12.]

[2] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022. [Pages xi, 1, 13, and 14.]

[3] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," 2022. [Pages 2, 13, and 56.]

[4] V. D. Lai, N. T. Ngo, A. P. B. Veyseh, H. Man, F. Dernoncourt, T. Bui, and T. H. Nguyen, "Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning," *ArXiv*, vol. abs/2304.05613, 2023. doi: 10.48550/arXiv.2304.05613 [Page 2.]

[5] Y. Deng, W. Zhang, S. J. Pan, and L. Bing, "Multilingual jailbreak challenges in large language models," *ArXiv*, vol. abs/2310.06474, 2023. doi: 10.48550/arXiv.2310.06474 [Page 2.]

[6] A. Ekgren, A. C. Gyllensten, F. Stollenwerk, J. Öhman, T. Isbister, E. Gogoulou, F. Carlsson, A. Heiman, J. Casademont, and M. Sahlgren, "Gpt-sw3: An autoregressive language model for the nordic languages," *arXiv preprint arXiv:2305.12987*, 2023. [Pages 2, 12, and 32.]

[7] J. Öhman, S. Verlinden, A. Ekgren, A. C. Gyllensten, T. Isbister, E. Gogoulou, F. Carlsson, and M. Sahlgren, "The nordic pile: A 1.2tb nordic dataset for language modeling," 2023. [Pages 2 and 12.]

[8] IBM. (2024) What is NLP? [Online]. Available: https://www.ibm.com/topics/natural-language-processing [Page 6.]

[9] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2024. [Page 7.]

[10] Y. Goldberg, "A primer on neural network models for natural language processing," 2015. [Pages 7 and 8.]

[11] J. J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*, 1992. [Online]. Available: https://aclanthology.org/C92-4173 [Page 9.]

[12] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018. doi: 10.18653/v1/P18-1007 pp. 66–75. [Online]. Available: https://aclanthology.org/P18-1007 [Page 9.]

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016. [Pages 9 and 10.]

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Pages xi, 9, 10, and 11.]

[15] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245 [Page 11.]

[16] F. Stollenwerk, "Training and evaluation of a multilingual tokenizer for gpt-sw3," 2023. [Page 13.]

[17] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," 2023. [Pages xii, 15, 16, 41, and 61.]

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Pages xii and 17.]

[19] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023. [Page 18.]

[20] Z. Guo, R. Jin, C. Liu, Y. Huang, D. Shi, Supryadi, L. Yu, Y. Liu, J. Li, B. Xiong, and D. Xiong, "Evaluating large language models: A comprehensive survey," 2023. [Pages xii, 18, and 19.]

[21] Hugging Face, "Open llm leaderboard," 2023, accessed: 2024-06-14. [Online]. Available: https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard [Page 19.]

[22] D. Nielsen, "ScandEval: A benchmark for Scandinavian natural language processing," in *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, T. Alumäe and M. Fishel, Eds. Tórshavn, Faroe Islands: University of Tartu Library, May 2023, pp. 185–201. [Online]. Available: https://aclanthology.org/2023.nodalida-1.20 [Pages 19, 34, and 55.]

[23] N. Alzahrani, H. A. Alyahya, Y. Alnumay, S. Alrashed, S. Alsubaie, Y. Almushaykeh, F. Mirza, N. Alotaibi, N. Altwairesh, A. Alowisheq, M. S. Bari, and H. Khan, "When benchmarks are targets: Revealing the sensitivity of large language model leaderboards," 2024. [Page 19.]

[24] O. Holmström and E. Doostmohammadi, "Making instruction finetuning accessible to non-English languages: A case study on Swedish models," in *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, T. Alumäe and M. Fishel, Eds. Tórshavn, Faroe Islands: University of Tartu Library, May 2023, pp. 634–642. [Online]. Available: https://aclanthology.org/2023.nodalida-1.62 [Page 21.]

[25] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. (2023) Free dolly: Introducing the world's first truly open instruction-tuned llm. [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm [Page 24.]

[26] AI Sweden, "GPT-SW3-6.7B-v2-translator," 2024. [Online]. Available: https://huggingface.co/AI-Sweden-Models/gpt-sw3-6.7b-v2-translator [Page 25.]

[27] D. Smilkov, "Lilac: Curate Better Data for LLMs," 2024, GitHub repository, https://github.com/lilacai/lilac (commit: b7d92b7). Accessed: 2024-06-26 . [Page 26.]

[28] V. Shepelev, "Spylls: Hunspell ported to Python," 2024, GitHub repository, https://github.com/zverok/spylls (commit: 9a0d201). Accessed: 2024-06-28 . [Page 28.]

[29] C. de Dampierre, "BunkaTopics," 2024, GitHub repository, https://github.com/charlesdedampierre/BunkaTopics (commit: ca11546). Accessed: 2024-06-28 . [Page 30.]

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703 [Page 33.]

[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020. [Online]. Available: https://arxiv.org/abs/1910.03771 [Page 33.]

[32] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022. [Page 33.]

[33] AI Sweden, "BiaSWE," 2024, HuggingFace Dataset. [Online]. Available: https://huggingface.co/datasets/AI-Sweden-Models/BiaSWE [Page 36.]

[34] W. Lian, G. Wang, B. Goodson, E. Pentland, A. Cook, C. Vong, and "Teknium", "Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification," 2023. [Online]. Available: https://https://huggingface.co/Open-Orca/SlimOrca [Page 72.]

[35] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah, "Orca: Progressive learning from complex explanation traces of gpt-4," 2023. [Page 72.]

[36] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for "mind" exploration of large scale language model society," 2023. [Page 72.]

[37] L. Daniele and Suphavadeeprasit, "Amplify-instruct: Synthetically generated diverse multi-turn conversations for efficient llm training." *arXiv preprint arXiv:(coming soon)*, 2023. [Online]. Available: https://huggingface.co/datasets/LDJnr/Capybara [Page 72.]

Appendix A

Supporting materials

A.1 Synthetic Data Generation

```
You are to generate instruction-tuning data in Swedish. The
questions and answers should be as elaborate as possible. Not every
question has to be formulated as an actual question with a question
mark, it can also be formulated as a statement or a prompt - be
creative.

Also think about adding some diversity to how the questions are
started, for example questions starting with ”Varför” or ”När”
or ”Jag” or ”Vilken” or ”Hur” or ”Kan” or ”Gör”, or ”Jag har en
fråga”, or ”Jag har ett problem”, or ”Fråga:”, or ”Jag vet inte..”,
or ”Jämför”, or ”Hjälp mig” , or ”Är det”, or ”Är”, or ”Nu”. Point
is; keep it diverse, think about formulating them in a diverse
number of ways so that there are a lot of different formats. Also,
the prompting can be more than just questions, it can be prompts
to list something in a specific order, a pop quiz, or any other task
you can think of.

If you want to add new lines in the questions or the answers, then
add a newline character in the string, but don’t actually create a
new line. However, this does not have to occur in every example, or
even any, but do it where it seems necessary.

The format should be in JSONL format like this:

{”text”:[”<human>”:”question”,”<bot>”:”answer”]}

Let’s start. Generate 20 examples in swedish about pronounciation
of swedish words.
```

Figure A.1: One example of how prompting was done with ChatGPT-4 to generate 20 examples regarding pronunciation of Swedish words.

```
{"text":["<human>":"Vad är det korrekta uttalet av
'kex'?","<bot>":"Uttal av 'kex' kan variera regionalt i Sverige. I
de flesta delar av landet uttalas det med ett hårt 'k'-ljud följt av
ett kort 'e' och ett 'x' som uttalas som ett 'ks'-ljud, så /kɛks/.
Dock, i vissa delar, som i Göteborg, kan det uttalas som /çɛks/ med
ett inledande 'sch'-ljud."]}
```


Figure A.2: One of 20 examples generated from the prompt given in Figure A.1.

A.2 Dataset Breakdown

Below lists details about the datasets collected, in the order presented in Table 4.1.

SlimOrca-SV

SlimOrca is a small subset of the OpenOrca dataset [34, 35]. This is a large dataset with curated instruction data, with the Slim version incorporating additional curation methods. This dataset contains system prompts.

CamelAI-SV

CamelAI Domain Datasets is a collection of datasets from CamelAI that contains instruction data with domain expertise in Physics, Math, Chemistry & Biology. The data typically consists of logic and problem solving questions in a chain-of-thought manner [36].

Pure-Dove-SV

The Pure-Dove dataset contains back-and-forth conversations between the human and assistant [37]. This is in contrast to most other datasets, which typically only contains one conversation turn per example.

OpenHermes-SV

OpenHermes is a large collection of different instruction datasets, e.g. CamelAI and SlimOrca amongst others. Although a smaller subset of the dataset was translated, an attempt was made to keep the distribution of the datasets the same.

Orca-DPO-Pairs-SV

This is a dataset with DPO-pairs, derived from the OpenOrca dataset. It also contains system prompts.

BibblanSvarar

Bibblansvarar was an initiative from the libraries of Malmö, Sweden, that from the assignment of Kungliga Biblioteket had a service where people could send in questions and get answers. This Question & Answer service was decided to be discontinued the spring of 2024. The dataset was curated and cleaned to fit an instruction format.

HP-ORD

This is a dataset that contains words from the ORD part of the Swedish SAT:s. Given a word x and its meaning y, the examples were formatted as a question ”What does x mean”, along with the answer ”x means y”.

BibblanSvarar-Synthetic

This is a synthetically generated dataset based on the questions that exist in the BibblanSvarar dataset. It was generated using the Llama-8B-Instruct model.

swedish-instruct-data-chatgpt4

This small synthetic instruction dataset contains question-answer pairs in Swedish that highlight a wide range of topics related to Sweden. It was generated using ChatGPT-4.

BezzerWizzer

Bezzerwizzer is a Swedish question/answer board game. This dataset was generated by giving a small part of the questions to ChatGPT-4, and manually fact checking the answers generated.

TRITA – EECS-EX 2024:0000
Stockholm, Sweden 2024

www.kth.se

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Olsén”,
”First name”: ”Tim”,
”Local User Id”: ”u100001”,
”E-mail”: ”timolsenkth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Cycle”: ”2”,
”Course code”: ”DA231X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Degree Programme in Computer Science and Engineering”
,”programcode”: ”CDATE”
,”Degree”: ”Degree of Master of Science in Engineering”
,”subjectArea”: ”Computer Science and Engineering”
},
”Title”: {
”Main title”: ”Designing a Pipeline for Creating and Evaluating Swedish Instruction Datasets for Large Language Models”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Formulering av en pipeline för att skapa och utvärdera svensk instruktionsdata för stora språkmodeller”,
”Language”: ”swe”
},
”Supervisor1”: { ”Last name”: ”Engwall”,
”First name”: ”Olov”,
”Local User Id”: ”u100003”,
”E-mail”: ”engwall@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Supervisor2”: { ”Last name”: ”Ekgren”,
”First name”: ”Ariel”,
”E-mail”: ”ariel.ekgren@ai.se”,
”Other organisation”: ”AI Sweden”
},
”Examiner1”: { ”Last name”: ”Gustafsson”,
”First name”: ”Joakim”,
”Local User Id”: ”u1d13i2c”,
”E-mail”: ”jkgu@kth.se”,
”organisation”: {”L1”: ””,
”L2”: ”Division of Speech, Music and Hearing” }
},
”Cooperation”: { ”Partner_name”: ”AI Sweden”},
"National Subject Categories": "10201, 10206",
”Other information”: {”Year”: ”2024”, ”Number of pages”: ”1,75”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA – EECS-EX” , ”No. in series”: ”2024:0000” },
”Opponents”: { ”Name”: ”A. B. Normal & A. X. E. Normalè”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
"Number of lang instances": "2",
”Abstract[eng]”: €€€€

```
\input{report/abstract-en}
```

€€€€,
”Keywords[eng]”: €€€€
Swedish Instruction Data, Model Fine-Tuning, Instruction Fine-Tuning, GPT, Large Language Model, Natural Language Processing, Artificial Intelligence €€€€,
”Abstract[swe]”: €€€€

```
\input{report/abstract-sv}
```

€€€€,
”Keywords[swe]”: €€€€
Svensk Instruktionsdata, Modellfinjustering, Instruktionsfinjustering, GPT, Stor Språkmodell, Naturlig Språkbehandling, Artificiell Intelligens
€€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                      or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% example of putting in a trademark on first expansion
\newacronym[first={NVIDIA OpenSHMEM Library (NVSHMEM\texttrademark)}]{NVSHMEM}{NVSHMEM}{NVIDIA OpenSHMEM Library}

\newacronym[shortplural={LLMs}, firstplural={Large Language Models}]{LLM}{LLM}{Large Language Model}

\newacronym[shortplural={LMs}, firstplural={Language Models}]{LM}{LM}{Language Model}

\newacronym[shortplural={GPTs}, firstplural={Generative Pre-Trained Transformers}]{GPT}{GPT}{Generative Pre-Trained Transformer}

\newacronym[]{GPT-3}{GPT-3}{Generative Pre-Trained Transformer 3}
\newacronym[]{RLHF}{RLHF}{Reinforcement Learning from Human Feedback}

\newacronym[]{PEFT}{PEFT}{Parameter-Efficient Fine-Tuning}

\newacronym[]{LoRA}{LoRA}{Low-Rank Adaptation}
\newacronym[]{QLoRA}{QLoRA}{Quantized Low-Rank Adaptation}

\newacronym[]{PPO}{PPO}{Proximal Policy Optimization}
\newacronym[]{DPO}{DPO}{Direct Preference Optimization}


\newacronym[shortplural={ANNs}, firstplural={Artifical Neural Networks}]{ANN}{ANN}{Artificial Neural Network}
\newacronym[shortplural={FNNs}, firstplural={Feed Forward Neural Networks}]{FNN}{FNN}{Feed Forward Neural Network}
\newacronym[shortplural={RNNs}, firstplural={Recurrent Neural Networks}]{RNN}{RNN}{Recurrent Neural Network}

\newacronym[shortplural={NLPs}, firstplural={Natural Language Processing}]{NLP}{NLP}{Natural Language Processing}
\newacronym{NLG}{NLG}{Natural Language Generation}
```