Master Thesis

Exploring Open-source Generative Models for Lexical Simplification through Prompt Learning

Swarupa Hardikar

a thesis submitted in partial fulfilment of the requirements for the degree of

MA Linguistics
(Text Mining)

Vrije Universiteit Amsterdam

Computational Lexicology and Terminology Lab
Department of Language and Communication
Faculty of Humanities

Supervised by: Dr. Luis Morgado da Costa, Dr. Hennie van der Vliet
2^{nd} reader: Dr. Piek Vossen

Submitted: 15 August, 2023

Abstract

Currently, there exists an emphasis to study the effect and performance of up-and-coming open-source generative models in undertaking various downstream natural language processing (NLP) tasks. On this basis, the current study leverages two such models, Alpaca (Stanford, based on LLaMA by Meta) and Orca (Microsoft Research), to build a prompt-based lexical simplification (LS) pipeline for the English language. Based on the TSAR-2022 Shared Task on Lexical Simplification [1], the study attempts to gauge the performance of these models and compare them against those of current state-of-the-art LS systems. Of the 20 experimental setups, two achieve near-state-of-the-art results, outperforming the task's upper benchmark in the process. It is found that Orca (13B parameters) takes well to contextual aid, while Alpaca (7B parameters) demonstrates comparatively limited performance with the same. Significantly, the research also reveals that the innate ability of the models to provide well-ranked substitutes given a complex word can eliminate the need for sophisticated ranking systems.

[1] https://taln.upf.edu/pages/tsar2022-st/

Declaration of Authorship

I, Swarupa Sharad Hardikar, declare that this thesis, titled *Exploring Open-source Generative Models for Lexical Simplification through Prompt Learning* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 15 August 2023

Signed: Swarupa Sharad Hardikar

Acknowledgments

I would like to extend my gratitude to my primary academic supervisor, Dr, Luis Morgado da Costa for taking me on as an advisee. Thank you for your expertise, impeccable guidance, advice and your patience. It has made the thesis period a very enlightening experience for me. I would also like to thank my secondary supervisor, Dr Hennie van der Vliet for his support.

I would like to thank the faculty at CLTL and my instructors for distributing their knowledge, enabling me to complete my education. Special acknowledgement goes to Dr Antske Fokkens, for her exemplary instruction and support, as well as for believing in me. It has kept me motivated for the entire duration of the project. I would also like to acknowledge Sal and Sofia for some key fact-checking, and the peers and friends I made along the way for their indispensable tips, encouragement and support.

I would like to thank my brother, Patanjali, for making this distant dream of studying computational linguistics here a reality. His encouragement of my interest in the discipline as well as his guidance have been invaluable. I would also like to thank my parents for their enduring support throughout my life, making me the person I am today.

List of Figures

Contents

Chapter 1

Introduction

1.1 Aim and Relevance

In a rapidly digitizing world, there remains an urgent need to improve communication such that any piece of information can be equally accessible to people of all groups. Due to the complexity of languages and domains, among other textual aspects, there is often a likelihood that some pieces of information seem elusive to certain people. This may be due to the complexity of subject matter, complexity of the language structure, or merely a word-level complexity.

It is well-known that proficiency in reading can vary from person to person, with some readers demonstrating more difficulty in doing so than others. For decades, this has led to the need for technological intervention to aid reading. (Alqahtani, 2020) Various efforts have been undertaken to bring about the ease of accessibility for readers, some more popular ones including Simple Wikipedia, which provides a shorter summary consisting of simpler words corresponding to a particular page, and Reddit's Explain Like I'm 5 (ELI5), a forum which explains terms, phenomena and the like a simplistic and non-academic way. On the other hand, automating such efforts has also been on the forefront within the field of natural language processing (NLP). It is thus imperative that efforts should be taken to further automate this process, and in the same vein, make advancements towards improving and expanding upon the same. This is where text simplification comes in.

1.1.1 Text Simplification

Text simplification is a popular task in NLP that involves simplifying the target text to enhance understanding for a certain target audience. It tackles the target text to reduce its complexity while retaining valuable information within it. Doing so can be a complicated effort, as retaining the nuance of a piece of text can be challenging.

Text simplification is divided into three levels of simplification: word level, sentence level, and document level. For the word level, it simply involves identifying complex words within the text and then replacing them with simpler words without changing the context. In theory, this process is implemented primarily on the lexical level. On the sentence level, there is more significant syntactic and semantic remodelling involved, such as restructuring sentences to make them shorter, breaking down complex sentence structures into simpler ones, and the like. Finally, text simplification on the document level accounts for the highest level of restructuring, including but not

limited to shortening paragraphs for a more bite-sized absorption of information or summarizing the text. For the purpose of this research, the scope of simplification is concentrated only on word-level text simplification, i.e., lexical simplification.

1.1.2 Lexical Simplification

Lexical simplification is a sub-task of text simplification, wherein complex words are replaced with their simpler substitutes in a piece of text for a variety of reasons. These may entail the ease of accessibility and readability for readers with low literacy/non-native proficiency, children, or non-subject matter experts.

1.2 Scope of the Research

The thesis closely follows the TSAR-2022 Task for lexical simplification for improved readability (Saggion et al., 2023). The TSAR-2022 was a multilingual shared task held as a workshop in conjunction with EMNLP-2022 for the following languages: English, Spanish and Portuguese. For the scope of this project, I will be considering only the English Language. The project keeps in close focus the winners and top performers of the shared task - University of Heidelberg (UniHD) and University of Manchester & Manchester Metropolitan University (UoM&MMU) - having utilised various models (including but not limited to BERT and its variants, GPT-3, and the like) to test their performance for the task of lexical simplification.

For the purpose of the shared task, the evaluation metric with which every system was compared and ranked was ACC@1. UoM and MMU's system achieved an ACC@1 of 0.6353, while UniHD scored the highest at ACC@1 0.8096. ACC@1 is the same as Potential@1, MAP@1 and Precision@1, and it is the primary metric according to which the Shared Task ranked entries. These metrics will be explained and discussed further in the following chapters. (Saggion et al., 2023)

The winner of the Shared Task made use of GPT-3's text-davinci. Considering the fact that GPT-3 is closed-source, this raises the question of how other open-source generative models, such as LLaMA (a recently released model by Meta) [1] Touvron et al. (2023) and GPT-2 (also the predecessor of GPT-3) (Radford et al., 2019), perform. In the same vein, it would also be interesting to compare them to the performance of current LS benchmarks such as LSBERT. One could even put to test how recent state-of-the-art generative models such as Orca (Mukherjee et al., 2023), also a large language model recently released by Microsoft Research, compare as well.

As mentioned above, for the UniHD entry, Aumiller and Gertz (2022) attempt a highly straightforward system of **prompting** the LLM (GPT-3 in this case) to generate simpler substitutes for the complex word, with or without context (i.e., the sentence from which the complex word originates). UniHD did not create separate system components to select and/or rank the substitutes, basing this on the assumption that the model is wholly capable of appropriately ranking the substitutes from most to least relevant and simple.

Prompt learning involves leveraging the knowledge of a language model by giving it the input of a specific prompt and having it generate a corresponding output. Designing a prompt can be done in two ways, namely engineering it or generating it.

[1] https://ai.meta.com/llama/

UniHD (Aumiller and Gertz, 2022) implemented the same process, wherein it conducted multiple experiments based on the variation of different prompts and their settings (i.e., zero, single or few-shot). They also attempted an ensemble prompt technique combining all outputs, which resulted in the winning run. However, it is worth noting that the model (as of June 2023) is a closed-source language model. The usage of such proprietary material may not be fully conducive to the norm of accessible and free-to-all research. As such, the primary objective is to make use of open-source technologies to advance research in this particular topic within NLP.

For the UoM entry, Vásquez-Rodríguez et al. (2022), used BERT's masked language modelling to inject prompts into the context for the model to generate word substitutions. The UoM authors set about designing a template for the prompting, such as "A(n) (Prompt1) (Prompt2) for (target-word)". After multiple experiments with prompting, such as using "easier" vs "simpler", they decided on "a simple word for X is [MASK] ." where X stands for the complex word.

For the purpose of this project, following the above experiments, I want to see how using generative models for prompting could change the outcome of the results.

1.3 Goal and research questions

1.3.1 Research Question

To go forward with the above-mentioned plan, I propose testing open-source large language models such as GPT-2 and LLAMA on the task of lexical simplification and carefully analyzing their performance.

Primary Research Question: Can one successfully leverage open-source models for prompt learning to build a lexical simplification system for the English language?

Sub Questions:

- How does the performance of these generative models compare against each other?
- How does their performance compare against that of the current state-of-the-art lexical simplification systems?

1.4 Pipeline

A typical lexical simplification pipeline consists of four major steps: Identification of complex words (CWI), Generation of simpler substitute words (SG), Selection of substitutes (SS), and Ranking of substitutes (SR). A fifth step, often optional depending on the language of the task, is the morphological generation and contextual adaptation of the substitutes (Paetzold and Specia, 2017a; Saggion et al., 2023). A more comprehensive summary of these steps can be found in the next chapter.

The very first step of the task, CWI, involves the crucial step of identifying the complex word(s) within the given sentence so that substitutes may be generated in place of them. It should be noted that this component has already been provided by the Shared Task as part of a testing dataset. Hence, I will not focus on its implementation. Additionally, as English is not a language that requires significant morphological adaptation given the context, the fifth step is also dropped with the assumption that the model can generate substitutes that won't require such adaptation. Briefly explained below are the three **primary** steps that the project focuses on:

1. Generation of simpler substitute words (SG): Given a complex word, what simpler substitutes can one think of? This is what the large language generative model tackles, i.e., generating substitutes for the word. This is where I would like to utilize the prompt learning approach of Aumiller and Gertz (2022), by prompting the model for easier words given the context (or without it) and generating substitute words.

2. Selection of substitutes (SS): This step tackles the selection of the appropriate k-substitutes for all the substitutes generated in the previous step. Which words are relevant to the complex word and make sense as substitutes, and which are completely irrelevant or just plain noise? For a zero-shot setting, gauging the appropriateness could be done with various approaches such as word embeddings or similarity. In the case of UniHD (Aumiller and Gertz, 2022), SS is delegated to the generation process itself.

3. Ranking of substitutes (SR): The penultimate step is to rank the appropriateness (in this context, simplicity) of each substitute word selected for the complex word. It has multiple approaches, including but not restricted to binary classification, cosine similarity, and fine-tuned masked language modelling (MLM) classification. Much like SS, SR can also be delegated to the generation process, as implemented by (Aumiller and Gertz, 2022)

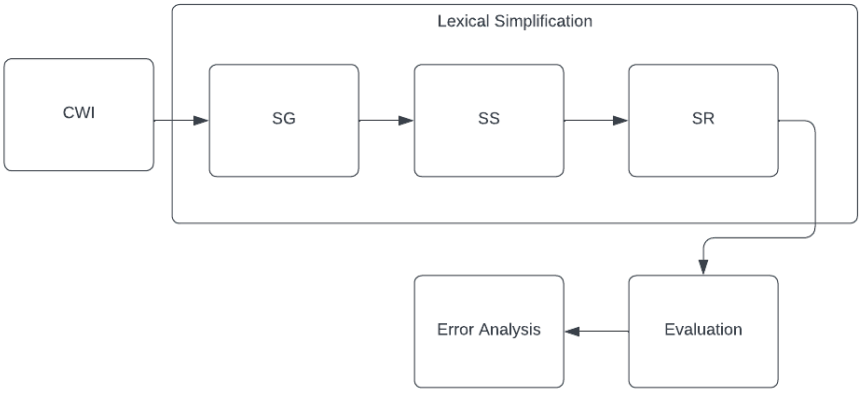

Figure 1.1: Proposed Pipeline

1.5 Data

The testing dataset has also been provided by the shared task. It is distributed into trial and test datasets; the latter consists of 370+ instances. The trial data does not have much to offer at only 10 instances. The format roughly consists of the sentence, the complex word within the sentence, and a number of gold-ranked substitutions for the same. FOr every data file containing the gold labels, there is one that is unlabeled, for

Sentence	Complex Word	Sub1	Sub2	Sub3
That prompted the military to deploy its largest warship, the BRP Gregorio del Pilar, which was recently acquired from the United States.	deploy	send	post	use

Table 1.1: Sample data snippet (Saggion et al., 2023)

ease of conducting the experiments. The data is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

The data follows the same format as most publicly available LS datasets, which includes the sentence, complex word within the sentence needing replacement, and replacement substitutes. Shown below is an example (the number of substitutes varies per instance:

(sentence), (complex word), (sub1), (sub2), (sub3)

Here is an example of an instance from the trial data (Saggion et al., 2023)

Sentence: A Spanish government source, however, later said that banks able to cover by themselves losses on their toxic property assets will not be forced to remove them from their books while it will be compulsory for those receiving public help.

Complex word: compulsory

Substitutes: mandatory mandatory mandatory mandatory mandatory mandatory mandatory mandatory mandatory mandatory mandatory required required required required required required required required essential forced important manadatory necessary obligatory unavoidable

The data is stored within tab-separated value (TSV) files and is uploaded on the Shared Task's official GitHub repository [2]. Further description of the data as well as its annotation can be found in Chapter 3.

1.6 Evaluation

The project adheres to the evaluation metrics of the shared task, mentioned as follows:

- Potential@k
- Accuracy@k@top1
- MAP@k

A further explanation of the evaluation metrics can be found in Chapter 3.

1.7 Outline

The chapters following this introduction have been highlighted as follows:

[2]https://github.com/LaSTUS-TALN-UPF/TSAR-2022-Shared-Task/tree/main/datasets

In Chapter 2, the literature relevant to this task will be explored, including a short description of the shared task, some notable entries of the task, and a brief introduction of the models. Chapter 3 will give an overview of the particulars of the Shared Task, such as its data (along with details on its procurement and annotation), evaluation process and baselines. Chapter 4 will provide an elaboration of the pipeline as a step-by-step walk-through of the following process. Chapter 5 will concentrate on the experiments performed: This will include a description of the utilization of the models and code as well as an overview of the results. Chapter 6 consists of an error analysis to go hand-in-hand with the results and provide possible explanations of the failures. Finally, Chapter 7 will consist of the discussion (including limitations and future work), and Chapter 8 will contain the conclusion.

Chapter 2

Review of the Relevant Literature

Over the years, the field of Lexical Simplification has seen a great amount of advancements in technology and methodology. From deriving word synonyms via lexical databases to using neural networks, the current chapter provides a summary of the development of LS systems. The chapter also offers an introduction of the Shared Task as well as a couple of participant systems that have inspired this project. This is followed by an overview of the models to be utilised for the project.

2.1 A History of Automatic Lexical Simplification

As highlighted by Paetzold and Specia (2017b), research on lexical simplification (LS) started gaining momentum a couple of decades ago, when Devlin and Tait (1998) extracted synonyms via WordNet, a large lexical database developed by Princeton (Miller, 1998) [1], which were then ranked as per the Kucera-Francis coefficient [2] . In doing so, the word with the highest ranking was chosen to replace the complex word. Kauchak and Barzilay (2006) take this a step ahead by involving both lexical and syntactic information in order to break down more complex sentence structures into simpler ones.

Paetzold and Specia (2017b) also highlight the original skeletal pipeline of the task, consisting of CWI, SG, SS and SR, attributing the same to Shardlow (2014b). The morphological adaptation of the substitutes is not discussed.

Although WordNet was used frequently in previous experiments, it was found that WordNet could not make for a full-proof solution for substitute generation (SG). Shardlow (2014) found that its limited vocabulary within the English language made for a drawback in generating enough simpler substitutes for a proposed word, causing erroneous generations and thereby warping the meaning of the sentences. Various other approaches have been utilized in order to automate substitute generation, including but not limited to parallel corpora (Biran et al., 2011), dictionary definitions (Kajiwara et al., 2013), and continuous bag-of-words (CBOW) via word2vec [3].

The process of substitute selection (SS) has also seen a lot of change throughout the years, starting from a "selection of all words" approach (Devlin & Tait, 1998; as

[1] //wordnet.princeton.edu/

[2] https://link.springer.com/content/pdf/10.3758/BF03204543.pdf

[3] https://www.tensorflow.org/text/tutorials/word2vec

quoted by Paetzold and Specia (2017b)) which reportedly led to a meaning change within a third of the instances. This was followed by attempts at modelling SS as a word sense disambiguation (WSD) task (Thomas and Anderson, 2012), part-of-speech filtering (selecting words with only the same POS-tags) (Aluísio and Gasperin, 2010) and by calculating a semantic similarity metric (Biran et al., 2011). The latter employed an ordered word pair of the complex word and a simplified word, and created a score indicating the cosine similarity between the two words.

Finally, Paetzold and Specia (2017b) list down the most frequently used approaches towards substitute ranking (SR), which are frequency-based (the intuition that frequent words are those more likely to be understood well), based on the measure of simplicity (involving creating a metric to calculate simplicity), and machine learning-assisted (i.e., algorithms such as "standard SVMs, Decision Trees and neural networks trained over lexical, morphological, psycholinguistic and semantic features") (Paetzold and Specia, 2017b).

2.2 LSBert

One of the foremost approaches towards using lexical simplification in line with transformer architecture came in the form of LSBert, as developed by Qiang et al. (2021). First introduced by Devlin et al. (2019), BERT (Bidirectional Encoder Representations from Transformers) is a language representation model that utilizes a masked language modelling (MLM) objective. Because of this approach, BERT is unique in that it tackles the unidirectionality of left-to-right architectures seen in previous models.

At its very base, the **initial** version of lexical simplification typically consisted of a three-step pipeline:

- Identification of complex words
- Generation of simpler substitute words
- Cleaning and ranking of the substitutes

With LSBert, Qiang et al. (2021) developed a framework that could encompass all three of the above steps while taking in their context - as opposed to previous iterations, which only considered the context in the last step, i.e., substitute ranking. For LSBert, the authors used a sequence-labelling method consisting of a BiLSTM (Bi-directional Long Short Term Memory) model to identify the complex word. They then used BERT to produce an appropriate list of substitute words for the complex word. This was done by masking the complex word of the original sentence as a new sentence, and then concatenating both sentences (the original and the new) for feeding to the BERT. The words with the highest probabilities were then chosen as substitute words. Finally, in order to rank the substitute words, the authors used five features: word frequency and word similarity, BERT's prediction order, BERT-based language model, and the paraphrase database (PPDB).

Thus, LSBERT simplifies complex words one at a time by considering the word complexity in context. The authors found that it achieved favourable evaluation outcomes with respect to three benchmarks: substitute candidates' evaluation, evaluation of substitute generation and ranking, and ultimately, evaluation of the entire system for sentence simplification. Across 3 well-known LS datasets (LexMTurk, BenchLS and

NNSEval), LSBERT achieved the highest accuracy scores (0.8, 0.6, and 0.4), respectively, outperforming the former state-of-the-art LS system (Paetzold-NE). Its net gain over the same is 29.8% (Qiang et al., 2021)

2.3 The TSAR 2022 Shared Task

The majority of this project is inspired by the TSAR 2022 task on Multilingual Lexical Simplification, which was included in the Workshop on Text Simplification, Accessibility, and Readability. The task sought to advance efforts towards lexical simplification in three languages: English, Portuguese and Spanish. The proposed pipeline for the task was roughly the same as the one given above, with the optional step of morphological generation and contextual adaptation.

The task notably had its own unique set of evaluation metrics to be used by each member team, consisting of:

- Potential@k (the percentage of instances for which at least one of the k top-ranked substitutes is also present in the gold data.)
- Accuracy@k@top1 (the percentage of instances where at least one of the k top-ranked substitutes matches the most frequently suggested synonym in the gold data)
- MAP@k (takes into account the position of the relevant substitutes among the first k-generated candidates (i.e., whether or not the relevant candidates are at the top positions)).

The next chapter goes into further detail about the above metrics.

The shared task featured two baselines for the evaluation of the models. The upper baseline consisted of LSBERT (described above) as the more competitive neural network-based approach to lexical simplification. The lower baseline consisted of a much simpler (dictionary-based) pipe-lined structure based on TUNER (Stajner et al., 2022), based on static resources, i.e., vocabularies and thesauri.

The English track of the shared task had the highest number of participating runs at 31. Out of these, only four runs (by three participants; mentioned below) outperformed the LSBERT baseline with respect to the ACC@1 metric (Saggion et al., 2023). For the purpose of generation, the majority of systems utilized neural networks (most of them being a version of LMs). Prompting was used only by Uom and UniHD, with the latter being the only participant using a generative LLM. For ranking, various techniques were employed, including but not limited to cosine similarity (which was used the most), BERT classifiers, word frequency, knowledge graphs, POS (part of speech) tag checks, and LM (language model) probability.

As mentioned previously, the winner for the task within the English language category was the entry by Heidelberg University (UniHD), which employed prompt learning for the state-of-the-art model GPT-3 - to be specific, text-davinci. It yielded the top two positions in the ranking list, with 2 different runs. (This was followed by MANTIS, which built upon and expanded the LSBERT model (see above) with increased accuracy. Finally, the third were the University of Manchester and Manchester Metropolitan University, who also used the approach of prompt learning to build their substitute generation model.

Team	Run	ACC@1	ACC@1@Top1	ACC@2@Top1	ACC@3@Top1	MAP@3	MAP@5	MAP@10	Potential@3	Potential@5	Potential@10
UniHD	2	0.8096	0.4289	0.6112	0.6863	0.5834	0.4491	0.2812	0.9624	0.9812	0.9946
UniHD	1	0.7721	0.4262	0.5335	0.5710	0.5090	0.3653	0.2092	0.8900	0.9302	0.9436
MANTIS	1	0.6568	0.3190	0.4504	0.5388	0.4730	0.3599	0.2193	0.8766	0.9463	0.9785
UoM&MMU	1	0.6353	0.2895	0.4530	0.5308	0.4244	0.3173	0.1951	0.8739	0.9115	0.9490
LSBert-baseline	1	0.5978	0.3029	0.4450	0.5308	0.4079	0.2957	0.1755	0.8230	0.8766	0.9463 height

Table 2.1: Results of the systems outperforming LSBERT

2.4 UniHD

As mentioned in the previous chapter, the winner of the TSAR-2022 Shared Task was the entry by University of Heidelberg (UniHD), presented by Aumiller and Gertz (2022) as well as the primary inspiration for the project. The authors mention their approach as "frustratingly simple", consisting of a pipeline based on prompted GPT-3 responses. This stems primarily from their assertion that complex pipelines with lots of elements are not always imperative for achieving exemplary, state-of-the-art results. On the same basis, they propose a much simpler one, as described below.

The English track of their submission includes a collection of six prompt templates with different levels of context. Some examples are shown below:

- *Context: {context_sentence}*
  temperature: Question: Given the above context, list ten alternatives for "{complex_word}" that are easier to understand.
  Answer: }

- *Give me ten simplified synonyms for the following word: {complex_word}* (Aumiller and Gertz, 2022)

A language transfer technique is then applied to propagate this project for Spanish and Portuguese. This is done simple by introducing the name of the language within the prompt, such as writing *"list 10* ***Spanish*** *words"*. The authors incorporated a pipeline with few-shot prompts for a vLLM (very large language model) given basic instructions on simplification, along with only four hand-labelled instances.

With the public release of GPT-3 by OpenAI, there has been a flurry of its implementation for tasks pertaining to natural language generation. The vLLMs released, with their ability of zero-shot transfer, highlight their usefulness for low-resource tasks. On the basis of this premise, the authors attempted to utilise prompted responses from the models for the generation of simplified text.

The authors first used zero-shot prediction for generating substitutions, instead of providing templates to the system, as a baseline. The system is provided with the context and is made to generate 10 simpler substitutions for the complex word. In order to filter the substitute candidates and subsequently rank them, a list of filters was applied to the candidates - note that no explicit step was created to actually rank the substitutes, assuming that the model would rank them automatically. The filtering step consisted of parsing the answer string to remove noise, newlines, whitespace, and the like. In addition, infinitives as well as multi-word answers were also removed.

For the second run, a collection of predictions generated from the templates was used. This was done to ensure that suitable predictions did not get discarded by the filtering system. The prompt templates were as follows: Zero-shot with and without context, Single-shot with and without context, and two-shot with context. This consisted of an ensemble run wherein answers were combined together from all of the

above-mentioned prompt templates and then re-ranked according to a combination score formula (Aumiller and Gertz, 2022)

The experiments provide excellent result metrics, with the first (single) and second (ensemble) runs scoring ACC@1 scores of 0.77 and 0.8 respectively. Most of the other metrics (Accuracy@k@Top1, MAP@k) of both runs also beat those of the other competitors. The authors also acknowledged many drawbacks within the experiments, such as unstable prompts, lack of context, and hallucinations. Further, they also acknowledged the enormous computing costs of such a project, which they bypassed by utilizing a paid API for GPT-3, aforementioned the proprietary model.

2.5 PromptLS

Another entry that should be studied for its use of prompting, although it did not directly involve the usage of generative language models, is the work of Vásquez-Rodríguez et al. (2022) for the TSAR 2022 task. The entry was the joint submission of the University of Manchester and Manchester Metropolitan University. The authors introduce the approach of prompt learning to the task of Lexical Simplification, wherein they use a predefined template within a masked language model (MLM) to generate substitute words.

The resultant model was called PromptLS, which utilized MLMs a little differently than its predecessor LSBERT: while the latter masked the token in context, PromptLS instead injected prompts (i.e., the predefined template mentioned above) to generate simpler substitutes for the given complex word. This was done by experimenting with different prompts to determine the best-performing one.

The final prompt - initially only in English - was then translated into Spanish and Portuguese to reproduce the results in those languages.

PromptLS consists of 3 modules: 1) an LLM to generate substitutes as per a given prompt; 2) a fine-tuned model building up on the previous one to select more appropriate words; and 3) a module to help delete erroneous/inappropriate candidates. As such, it contained two runs, one being a zero-shot approach and another being the fine-tuned LLM. For the fine-tuning, it made use of multiple lexical simplification datasets for each language, consisting of roughly the same format as the shared task's test dataset. Various MLM-based models were tested, including RoBERTa-large for English, BERTIN for Spanish, and BR_BERTo for Portuguese. A multilingual model mBERT was also used. Extensive candidate filtering was conducted as post-processing to remove unnecessary substitutes such as complex candidates and non-words generated by the models and duplicates. WordNet was also utilized for English to remove any antonyms.

The authors found that although the zero-shot approach worked well with English, this was not the case for the other two languages. Notably, it was also found that the multilingual model did not work as well as the monolingual ones. Fine-tuning also improved the performance of the models considerably, although the tuning data was not very big.

2.6 Models

An Introduction to Transformers Transformers, as introduced by Vaswani et al. (2023), consist of multilayer networks known as transformer blocks. These transformer blocks are made up of a combination of simple linear layers, feedforward networks,

and self-attention layers. Self-attention layers are important in that they allow the neural networks to "directly extract and use information from arbitrarily large contexts" (as opposed to recurrent neural networks, which require passing it via intermediate recurrent connections (Jurafsky and Martin, 2022). They map input sequences to same-length output sequences, where the model can access all of the inputs up to and including the current one. This enables their use for creating language models and utilising them for the purpose of autoregressive generation (explained below).

Generative LLMs The current project tests the performance of generative large language models on the task of lexical simplification. Generative language models are attention-based models with the ability to generate new text by continuing an initial word sequence, where the word to be generated can be located either at the end of the original sentence or in the middle of it. Unlike autoencoders such as BERT, generative models are autoregressive. The training of autoregressive models consists of using a sampling suite of words from the source dataset (known as "utterances"). Autoregressive models are also referred to as "auto-complete", as they are trained to complete sentences based on the datasets they are trained on. They are trained on vast amounts of text data and use probabilistic methods to predict the word most likely to occur next in a particular sequence (Kucharavy et al., 2023).

In order to start generating text, the models require a starting piece of text referred to as a "prompt". The models then look for the word that might best continue that prompt within their training datasets. In order to do so, a sampling strategy is required, which may be maximum likelihood, beaming, top-K, and top-p/nucleus.

The past few years have noticed a substantial increase in the size of generative models, which has been observed to be hand-in-hand with the increase in their performances. However, when considering training dataset sizes instead, it has been recently found that some smaller models can perform at par with larger-sized models as well (Kucharavy et al., 2023).

Below, the models to be utilized for the project are introduced.

2.6.1 GPT-2

The capacity (i.e., its size and number of parameters) of a language model is key to achieving the success of zero-shot task transfer. Hence, to enhance performance in a log-linear manner across tasks, it is natural to increase the capacity. On the basis of this idea, Radford et al. (2019) trained their models on a dataset of a million webpages (called WebText). The largest resultant model, GPT-2 (Generative Pre-Trained Transformer), consists of 1.5B parameters.

On being tested on eight language modelling datasets in a zero-shot setting, GPT-2 was able to achieve state-of-the-art results for seven of them.

The authors use language modelling as their primary approach, wherein the probabilities of word sequences are decided based on their frequency in a vast corpus of text. Considering the natural sequential ordering of human languages, one can factorise the "joint probabilities over symbols as the product of conditional probabilities" (Radford et al., 2019). The authors believe that language models with a sufficient capacity can learn to infer and perform the above-mentioned natural sequences and thus can better predict them. This is essentially in line with "unsupervised multitask learning". As

such, the authors are able to establish that such a model can use its general-language understanding to perform tasks without any need for task-specific data (Radford et al., 2019)

The models rely primarily on the well-established Transformer-based architectures (Vaswani et al., 2023) for the models, expanding upon and modifying the pre-existing GPT model by OpenAI.

The authors evaluated the model on a zero-shot setting on various popular NLP tasks, including but not limited to testing long-term dependencies (LAMBADA), named entities (The Children's Book Test, CBT), machine translation, and reading comprehension, to highly successful results. Some other tests, such as summarization, yielded less-than-satisfactory results. However, on an overall basis, the results implied the versatility of the models in being able to perform optimally in the case of various tasks without explicit supervision. It was also noted that it would be interesting to test the model out on more downstream NLP tasks for further expansion of research.

2.6.2 LLaMA

With the rapid rise in the development of large language models, researchers are currently keen on experimenting on increasing the performances of the same. Similarly, efforts are being undertaken to develop models such that they are as accessible for further research purposes as possible.

LLaMA, as developed by Touvron et al. (2023), is an attempt to implement the same. It is a collection of LLMs that range from 7B to 65B parameters. Trained only on public open-source datasets (consisting of 7 trillions of tokens), LLaMA has completely avoided using any proprietary and inaccessible datasets. This is in contrast to a number of mainstream models currently in use, such as GPT-3, PaLM, and Chinchilla, which makes LLaMA the ideal set of models for any open-source efforts in research. Some datasets used include English CommonCrawl, Github, Gutenberg, Wikipedia, ArXiv, etc.

The authors quote Hoffmann et al. (2022) in claiming that recent research has asserted that the number of parameters is not necessarily the only thing boosting the overall performance of LLMs, as had been thought previously. Instead of large models, it was found that small models that were trained on more data yielded the most optimal performances. This finding was a key ingredient with respect to the development of LLaMA. Its architecture was based on the traditional transformer architecture developed by Vaswani et al. (2023), with some tweaks inspired by other LLMs: Pre-normalization [GPT3], SwiGLU activation function [PaLM], Rotary Embeddings [GPTNeo].

A number of experiments were conducted on the models based on multiple NLP tasks, on zero-shot and few-shot settings. The models showed optimal performances, especially the 65B model, in Common Sense Reasoning (67.9% vs 58.4% (GPT-3) for RaceMiddle) and Question Answering (68.2% for TriviaQA), especially in comparison with GPT-3 and PaLM. It also had satisfactory results for Massive Multitask Language Understanding (MMLU) and Mathematical Reasoning (despite not being finetuned on mathematical data), being on a highly competitive level with PaLM and Minerva, respectively. The authors also found that finetuning the models on instructions yielded highly successful results (68.9%, although still lower than state-of-the-art - 77.4%).

LLaMA's open-source status as well as transparency with regard to training data, combined with its impressive performance metrics, makes it an ideal model for con-

ducting academic research.

The Self-Instruct Framework and Alpaca

The model used for the current project is Alpaca, which has been fine-tuned from LLaMA's 7B version on "52K instruction-following demonstrations" [4]. The authors use a self-instruct framework Wang et al. (2023) for the demonstrations generated using "text-davinci-003". Self-instruct involves enhancing the capabilities of pre-trained language models in regard to instruction-following by "bootstrapping off their own generations" (Wang et al., 2023). To elaborate, instructions, input, and output samples from a language model are first generated. Following this, invalid or similar samples are filtered out, and clean samples are then used to finetune the original model. The authors find that on the "SUPER-NATURAL INSTRUCTIONS" task, SELF-INSTRUCT shows a 33% absolute improvement in performance over the original GPT-3. SELF-INSTRUCT also outperforms various other GPT-3 variants trained on publicly available instruction datasets.

2.6.3 Orca

Currently, many NLP systems do not demonstrate the ability to conduct complex reasoning tasks and may not always be able to generalize to new domains. Consisting of 13 billion parameters, Orca is a language model built upon the LLaMA family of models Mukherjee et al. (2023) that attempts to address the abovementioned limitations.

The authors expand upon the current trend of research that seeks to mimic the behaviour of Large Foundation Models (LFMs) by training smaller models. This is done by learning to "imitate the reasoning process of" LFMs with the help of "complex explanation traces of GPT-4" (Mukherjee et al., 2023). Developed by Microsoft Research, Orca learns from *rich signals* from GPT-4. This mainly consists of explanation traces, step-by-step thought processes, and other complex instructions. The model is guided by teacher assistance from ChatGPT. Mukherjee et al. (2023) implement a technique called *explanation tuning*, wherein system instructions such as "explain like I'm five", "think step-by-step" and "justify your response" are used to yield explanations containing reasoning processes. Consequently, they believe that Orca's ability to learn from diverse and large-scale imitation data contributes to its positive results.

The authors utilise various evaluation processes to test Orca, including but not limited to the AGIEval (checks the model's ability to perform zero-shot reasoning) benchmark and the Big-Bench Hard benchmark (checks the model's ability of complex reasoning), respectively. For both, it (0.77 and 0.56 resp.) outperforms major state-of-the-art models such as GPT-3 (0.71 and 0.54 resp.). Mukherjee et al. (2023) also demonstrate the performance of Orca on several downstream tasks, including question answering, summarization, and dialogue generation. In all of these tasks, Orca outperforms other models, including GPT-3 and T5. Of note, Orca yields an accuracy of 0.77 on the SAT-English task, compared to 0.70 for GPT-3.

The authors conclude by stating the implication that smaller models can be trained to improve focus and be more adaptive in limited contexts without significantly compromising quality.

[4] https://crfm.stanford.edu/2023/03/13/alpaca.html

In summary, a comprehensive history of the lexical simplification task and its advancements through time are briefly explored. Information about state-of-the-art LS systems as well as the TSAR-2022 Shared Task on LS is highlighted. Finally, an introduction of the models to be utilised for the current project is provided.

Chapter 3

Data and Evaluation

As previously mentioned, this project is based on the TSAR 2022 Shared Task for Lexical Simplification (Saggion et al., 2023). Various components of the shared task, such as its rationale, data, evaluation, and top participants have been very briefly mentioned in Chapters 1 and 2. The current chapter provides further details for understanding three key aspects of the shared task, i.e., the data, the evaluation metrics, and the baselines.

3.1 Data

According to Saggion et al. (2023), the data, containing the sentences and their respective complex words, was procured from the BEA-2018 shared task on complex word identification (Yimam et al., 2018). Complex words were identified with the help of 10 native and 10 non-native speakers of English to prevent bias. Words were marked as complex when they were deigned to be "difficult to understand in a given context" by at least one annotator. Often, it was found that many sentences had more than one complex word needing simplification. For the purpose of the shared task, in such cases, only one complex word was chosen per sentence. In turn, this made it comparatively easier for the participants, as the factor of accounting for interactions between the chosen substitutes was no longer a problem.

For the 386 English instances, the crowdsourcing task was conducted on Amazon's Mechanical Turk, a well-known crowdsourcing marketplace [1]. The workers were given the task of suggesting a simpler substitute such that it would retain the meaning of the original sentence (Saggion et al., 2023). According to Štajner et al. (2022), the quality of crowdsourced substitutes was then checked by at least one computational linguist who was also a native speaker. The annotation guidelines were originally in Spanish (for Spanish data) but were translated for the English dataset with some editing to make sure they were consistent.

Finally, the annotated instances were split into trial and test datasets (10 and 373 respectively). Notably, the shared task did not provide any training data.

[1] https://www.mturk.com/

3.2 Evaluation

As mentioned in the introductory chapter, the evaluation criteria are provided by the Shared Task itself (Saggion et al., 2023). In the following sections, the significance of each of the evaluation metrics is discussed. Note that k denotes the k-top-ranked substitutes for each metric, where the values of k prominently used by the Shared Task are 1, 3, 5 and 10 (except Accuracy@k@top1, wherein the values of k are 1, 2 and 3. Given the fact that MAP@1, Potential@1, and Precision@1 would be the same as per their definition, they are combined together as ACC@1.(Saggion et al., 2023).

3.2.1 Potential@k

Potential@k denotes the percentage of instances for which at least one of the k-top-ranked substitutes is also present in the gold data. For example, given k=3, among the 3 substitutes *mandatory*, *required* and *essential* as generated by the system, if even one of these also shows up in the list of substitutes in the gold data, it increases the overall Potential@3.

This quantifier, especially on the higher ends (such as Potential@5 and Potential@10), is a weak one to gauge the performance of the model. It does not account for the position of the substitutes within the gold data. In regard to Potential@10, even one substitute that is possibly ranked last within the gold data could boost this metric for the system. For example, in an instance within the gold data, the word *teased* is the last-ranked substitute for the complex word *instilled*. Although it has somewhat of a relation with the original word, it is not necessarily the best when seeking an alternative for the word. As such, when the only match between the system-generated substitutes and the gold substitutes is *teased*, it can be considered the marker of a relatively poorer system. This makes Potential@k the most lenient quantifier of LS systems.

3.2.2 Accuracy@k@top1

Accuracy@k@top1 refers to the percentage of instances where at least one of the k-top-ranked substitutes matches the most frequently suggested synonym in the gold data.

This is the most punishing metric, being extremely sensitive to the position of the substitutes generated by the system (notably, the candidate in the first, second and third positions). Even if the top-ranked gold substitute exists within the list of system-generated substitutes (such as the fourth position), Accuracy@3@top1 would still penalize the system for that instance. As such, even if the model produces partially correct results, they too get penalised. Naturally, the criterion of Accuracy@1@top1 is even stricter. If the top-ranked (first candidate) substitute generated by the system is *mandatory*, and the top-ranked substitute within the gold data is the same, only then is the system rewarded for the instance. If *mandatory* is in the second position, however, the Accuracy@1@top1 score for the instance would be 0.

Considering this stringency, higher Accuracy@k@top1 scores are representative of a highly exemplary system that would always provide the best-simplified words first.

3.2.3 MAP@k

MAP@k stands for Mean Average Precision @ k, a popular metric prominently used for evaluating recommender systems which can also be applied to LS systems. It takes into

account the position of the relevant substitutes among the first k-generated candidates (i.e., whether or not the relevant candidates are at the top positions). Much like Accuracy@k@top1, it is also sensitive to the position of the substitutes within the gold data. However, it is far less punishing, given that instead of accounting for only the top candidates, it considers the whole ranking order of the system-generated substitutes. Thus, it provides a better, more in-depth evaluation of a given LS system.

In this case, the average precision (AP) of each *relevant* substitute is calculated, and then the mean of these scores is what is considered MAP@K. Consider the following example: for the substitutes *mandatory*, *required* and *difficult*, if the last is not a relevant substitute, this can lower the MAP@3 for the instance. Similarly, if *required* is not at the second position in the gold data, it can also lower MAP@3.

Note: To calculate MAP@k, Precision@k is also required, which is defined as the percentage of k-top-ranked substitutes that are also present in the gold data. Precision@k is not part of the major metrics displayed or *explicitly considered* by the Shared Task while calculating results.

3.3 Baselines

The Shared Task consisted of two baselines of varying performance levels: TSAR-TUNER on the lower end (ACC@1=0.3404)and TSAR-LSBERT (ACC@1=0.5978) on the higher end. For the English language, the two systems are derived from Stajner et al. (2022).

3.3.1 TSAR-TUNER

Originally an LS system for Spanish, TUNER is a proficient non-neural network that was adapted to work with English data for the purpose of the Shared Task. Developed by Ferrés et al. (2017), it consists of (1) Document Analysis, (2) Complex Word Detection, (3) Word Sense Disambiguation, (4) Synonyms Ranking, and (5) Language Realization. TSAR-TUNER has an extremely similar pipeline barring the complex word identification and context adaptation steps, which are not covered in the scope of the task. As can be gleaned from the above pipeline, TSAR-TUNER has a non-neural approach towards LS consisting mainly of databases and thesauri.

TSAR-TUNER yielded a middling performance on the Shared Task, with its ACC@1 being 0.3404. Surprisingly, as per Saggion et al. (2023), it still outperformed a few neural net-based systems in spite of not incorporating neural networks itself, which speaks well of its robustness. Regardless, it achieved the rank of 24th among the total 33 (including both baselines) English track runs. Nine systems performed worse than TSAR-TUNER.

3.3.2 TSAR-LSBERT

TSAR-LSBERT is developed from LSBERT, the transformer-based state-of-the-art system for LS in English (see Chapter 2 for more details). TSAR-LSBERT utilises the same resources as LSBERT for its pipeline (Saggion et al., 2023).

TSAR-LSBERT performed very well on the Shared Task, achieving an ACC@1 of 0.5978, the run yielding the 5th rank. Only four runs (by three participating teams) were able to outperform it.

In summary, the chapter provides a brief glance into the skeleton of the TSAR-2022 Shared Task, including details about its data (procurement, guidelines, and annotation), evaluation metrics and their significance w.r.t LS, and baselines.

Chapter 4

Method

The current chapter provides a description of the methodology utilised for the experimentation setup and a step-by-step walk-through of the pipeline as highlighted in the introductory chapter. The following methods are discussed: for substitution generation, the chosen generative models and prompts (along with their variations) are described. For the next step, i.e., substitute selection, the selection procedure (i.e., semantic similarity) and its model are described. For substitute ranking, the plan for a simple frequency based-ranking method using Google ngrams is laid out.

4.1 Prompt Engineering

Given that the scope of the thesis relies on examining the generating power of models to generate appropriate substitutes, producing prompts for the same is an integral step in this process.

The first approach to the experiment had been to replicate the prompts produced by Aumiller and Gertz (2022)(first place in English for the shared task) for a general idea of the response. Following this, I began to develop my own prompts (as highlighted in the next section). The idea is to test every prompt and manually gauge whether, for these prompts, the model is able to generate substitute words at all.

There are various modes of prompting, including zero-shot, one-shot and few-shot. In zero-shot prompting, the model is usually able to provide output for a prompt with no previous training based on the required task. On the other hand, few-shot prompting involves introducing a few examples relating to the task to the model for better adaptation of the task. For the purpose of this project, I will be focusing on two configurations: zero-shot with and without context (as highlighted above), along with one-shot prompts.

While it is likely that the model may take well to a certain prompt in the case of some instances, it may or may not be able to produce equally successful generations in the case of others. Keeping this in mind, continuous experimentation with multiple prompt templates needs to be implemented.

4.2 Substitute Generation

A key step to run a lexical simplification pipeline, substitute generation accounts for the greatest scope for the purpose of this project. In substitute generation, substitutes are generated for a target complex word within a sentence.

The motivation behind this step is to replace the complex word with a suitable simpler substitute such that it reduces the overall complexity of the sentence. As mentioned in the chapter, the process is implemented using our three models in a zero-shot setting in two ways: with and without the surrounding context. Naturally, the "without context" experiment consisted of making the model generate substitutes for the word without any information about the sentence whatsoever, while the context-containing experiment made use of the whole sentence containing the target complex word. To carry out these experiments, two prompt templates were created in English. For optimal results, 10 substitutes were requested per word. The prompt templates are as follows:

Context: "[sentence]" \n For the above context, **provide** 10 **easier** substitutes for the word "[complex word]"

Context: "[sentence]" \n For the above context, **list** 10 **easier** substitutes for the word "[complex word]""

The sentences and complex words for each instance within the dataset were plugged into each template accordingly.

As for the models, the aforementioned three, i.e., LLaMA, Orca and GPT-2 were used. For LLaMA, I used Alpaca-LoRA, a low-rank adaptation of Stanford's Alpaca which in turn is based on LLaMA. Alpaca is a language model that has been fine-tuned using LLaMA's 7B parameter variant to follow user instructions. In low-rank adaptations, the pre-trained model weights are frozen and trainable rank decomposition matrices are injected into each layer of the transformer architecture (Hu et al., 2021). This is known to reduce the number of trainable parameters for downstream tasks. In doing so, computational expenses are considerably reduced.

To run Alpaca-LoRA, I used a pre-existing Google Colaboratory [1] notebook demonstrating its usage with the 7B-parameter LLaMA. The configuration settings, which are commonly used to control the behaviour/response of the model, were set as per default. These are shown as follows:

- temperature=0.1: The temperature of a generative model determines the randomness of its responses. A higher temperature level (e.g. 1.0) can result in creative yet somewhat incorrect responses, while a lower temperature can give more focused but repetitive responses.

- top_p=0.75: Top-p is based on nucleus sampling, which shortlists the top tokens of which the sum of likelihoods does not exceed a certain value [2]. A higher top-p value yields more creative outputs.

- num_beams=4: By using beam search, the risk of missing hidden high-probability word sequences is averted by setting the number of beams of hypotheses at each time step. The hypothesis with the overall highest probability is finally chosen [3].

Orca was implemented using the "OpenAssistant/llama2-13b-orca-v2-8k-3166" by OpenAssistant, a chat assistant project that aims to make chat-based large language models accessible (Köpf et al., 2023). The model was made available using its Hugging Face model card [4].

[1] https://colab.research.google.com/drive/1eWAmesrW99p7e1nah5bipn0zikMb8XYC
[2] https://docs.cohere.com/docs/controlling-generation-with-top-k-top-p
[3] https://huggingface.co/blog/how-to-generate
[4] https://huggingface.co/OpenAssistant/llama2-13b-orca-v2-8k-3166

The prompt template for utilising the model is shown below:
"<prompter>{prompt}<endoftext><assistant>" [5]

4.2.1 Answer Processing

Generative models are built in a way that makes their output user-friendly and human-like to the user, which may entail more of a conversational tone to them. As such, the output more often than not does not always produce a straightforward answer (in this case, a list of substitutes for the required instance). In order to gain the required output, further filtering is required. Explained below are the techniques used to achieve this objective, highlighting the steps involved in processing the model's output to extract the required substitutes for a given complex word.

- Data Preprocessing

  The first step in the filtering process involves preprocessing the model-generated answer to remove any noise, such as introductory sentences and other extraneous parts. These might include phrases such as "Here are some easier words..." or "Given the context, some substitutes for (complex word) are..." To address this, the answer parser checks for the presence of a colon (':') within the answer, as this typically indicates the presence of structured information. Following these norms, the content following the last colon is selected. This step ensures that extraneous text before the actual answer is effectively filtered out. Some model-specific noise discovered during trial runs (such as some instances of the Orca run including postscripts such as "Note:..." are also eliminated.

- Data Cleaning

  After extracting the main content from the model's answer, the answer parser converts the text to lowercase. This transformation is essential to ensure that variations in capitalization do not influence the subsequent analysis, as the gold data substitutes are also lowercase. Lowercasing the text also assists in maintaining consistency throughout the dataset and simplifies word comparisons.

  The filtering process also aims to remove any words that do not significantly contribute to the semantic meaning of the response which may have escaped the post-colon parsing. This may also include bullet points or numbering used to list the substitutes. The answer parser removes any non-alphabetic characters (such as numerical or special characters other than hyphens) or whitespace. Consequently, only words remain in the response.

- Noise Removal

  It is also likely in the case that when a colon-based separation of the answer and its introduction is unavailable, various meaningless words may enter the list of substitutes. To ensure that any common stopwords (i.e., words that occur frequently but typically carry little semantic meaning, such as 'the', 'is', 'in', etc) are not included within the list of words, Natural Language Toolkit (NLTK) [6] is employed. NLTK, a package for NLP libraries and programs, contains a library

[5] As recommended by https://huggingface.co/TheBloke/OpenAssistant-Llama2-13B-Orca-v2-8K-3166-GGML

[6] https://www.nltk.org/

of such stopwords in the English language. Accordingly, stopwords are filtered out from the response to focus on potential substitutes.

In addition, to prevent the model from repeating the prompt initially created, a filter to remove the original prompt (or components from the original prompt) from the answer is also included.

Finally, before SS and SR, any duplicates that may have been generated by the model are also filtered out.

In this manner, the answer parser generates a list of cleaned substitutes. These cleaned substitutes can then be sorted according to the selection and ranking techniques mentioned below.

4.3 Substitute Selection

Substitute selection checks whether the substitutes generated for a given complex word are appropriate for the word and its context. In doing so, it is able to filter out words that are completely unrelated to the complex word; for example, if the model generates "gentle" and "calming" for the complex word "symphonic", such substitutes should be removed accordingly.

Various methods were initially tried for this step, including WordNet synsets as well as cosine similarity (using word embeddings). However, given that these techniques were context-agnostic, they were discarded. Finally, with the basis of contextual embeddings, I decided to use a sentence similarity model that compared the semantic similarity between two sentences and returned a similarity score.

The model in question is Sentence-BERT, a BERT variant developed by Reimers and Gurevych (2019). The developers note that the traditional BERT maps sentences to a vector space typically unsuitable for use with similarity measures such as cosine similarity. To improve upon this, Sentence-BERT utilises "siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity". In doing so, the model reduces the time taken to calculate similarity and effectively streamlines the effort.

SentenceBERT is utilized as follows: For each instance, one can replace the complex word with each substitute generated by the model, thereby creating n-sentences per number of substitutes. After this, each sentence can be compared against the root (original) sentence containing the complex word. Each sentence will then yield a similarity score in comparison to the root sentence. For example, consider an instance from the trial dataset (the complex word is highlighted in bold):

"A Spanish government source, however, later said that banks able to cover by themselves losses on their toxic property assets will not be forced to remove them from their books while it will be **compulsory** for those receiving public help."

In such a case, if the model generates the substitutes as "mandatory", "required" and "essential", one can create sentences out of the same to compare with the original sentence (mentioned above).

As for the actual substitute selection part, a threshold similarity score must be decided so that sentences (and consequently, substitutes) yielding scores lower than the same can be automatically filtered out. For example, as a preliminary test, the similarity score for the word "mandatory" came out to be 0.991 (with 1 being the highest) compared to the source sentence. That of the word "unavoidable" was slightly

lower at 0.983. From this experimentation, selecting a threshold score such as 0.90/0.95 could likely filter out any incompatible or highly irrelevant candidates.

Naturally, a simple guess such as the above cannot be a reliable basis for determining the threshold score, and further experimentation is certainly required. The most obvious way of doing so is naturally by trial and error. Unfortunately, given the size of the trial dataset provided by the shared task consisting only of 10 instances, other ways were needed to experiment with this. This was deduced by manually creating instances.

4.4 Substitute Ranking

Ensuring that the most suitable (and indeed, the easiest) substitutes are prioritised over the relatively less helpful ones is integral to the task of lexical simplification. Accordingly, one of the most important steps within the pipeline is substitute ranking.

As mentioned above, the method of choice for substitute selection was decided to be sentence similarity. Given the fact that the sentence similarity model produced similarity scores for each sentence compared to the source sentence, it was natural that such a method could also be used to calculate how well the word fits within the context. With this method in consideration, it was decided to use the similarity scores from highest to lowest in order to rank the substitutes. Thus, with a singular step, one can filter out the irrelevant/ill-fitting substitutes as well as rank the remaining ones according to the best fit.

It should, however, be noted that selecting this method for substitute ranking can have its drawbacks. A sentence similarity model will be able to rank substitutes only per the best fit according to the cosine similarity of word embeddings, which does not necessarily factor in the “simplicity” in the ranking process. This is an important point to consider, as the heart of the task is to simplify the sentence for the target reader.

As such, a good lexical simplification system should not only be context-aware but also be mindful of the level of simplicity required. One such approach relies on the theory that more frequently occurring words are more likely to be perceived as easier to understand. On the basis of this, it was decided to rank the substitutes according to their frequency based on Google ngrams (Michel et al., 2011). Google ngrams maintains an annual count of ngrams that have been documented from sources existing between 1500 and 2019. The online database is freely accessible and is able to measure the frequencies of any given term within its English corpus. The REST API makes use of Google’s Ngrams using the Google Books Ngram Dataset v3 [7].

Using the Ngrams API, it is possible to find out the frequency of a token based on the extensive Google ngram corpus. Since easier words are used more frequently by individuals, the frequency of the easiest substitute would likely be highest, and difficult words may be comparatively more infrequent. Consider an example from the trial data: for the complex word ”compulsory“, among two generated substitutes ”required“ and ”essential“, the frequency of ”required“ is 457,847,322, while that of ”essential“ is 137,850,046. According to the frequency theory, since ”required“ is used by individuals far more frequently than ”essential“, it is likely that the majority of individuals may find the former word easier.

With this logic, one can rank the simplicity of the substitutes from easiest to most difficult among the candidates generated by the models.

[7] https://github.com/ngrams-dev/

4.5 A More Straightforward Approach: The Superprompt

For the UniHD entry, Aumiller and Gertz (2022) did not implement any specific method for substitute selection and ranking and instead assumed that the model (GPT-3) will be capable of generating the resultant substitutes that were already ranked from easiest to most difficult. Drawing inspiration from this, I decided to experiment with a "superprompt" that made the model generate already-ranked substitutes without any further steps for ranking.

The idea is that a prompt can be created such that it fulfills all the major steps of the pipeline, i.e., SG, SS, and SR. The prompt was as follows:

"Context:'[SENTENCE]'\n Given the above context, list 10 easier substitutes for the word '[WORD]' Rank them from easiest to hardest."

It should be noted that the phrase "Given the above context, list" has been taken directly from a prompt created by Aumiller and Gertz (2022). The implementation of this variation is comparatively straightforward and easy compared to creating different systems for SG, SS and SR. Comparing this streamlined version to my main pipeline (as described in the above sections) is going to be a key point of comparison for this project.

In addition, for the one-shot prompt, the exact same prompt as UniHD was used (with the addition of "Rank them from easiest to hardest"). It is written as follows:

"Context: A local witness said a separate group of attackers disguised in burqas - the head-to-toe robes worn by conservative Afghan women - then tried to storm the compound.\n Question: Given the above context, list ten alternative words for "disguised" that are easier to understand.\n Answer:\n1. concealed\n2. dressed\n3. hidden\n4. camouflaged\n5. changed\n6. covered\n7. masked\n8. unrecognizable\n9. converted\n10. impersonated\n\n Context: '[SENTENCE]'\n Question: Given the above context, list ten alternatives for [WORD]' that are easier to understand. Rank them from easiest to hardest.\n"

As such, in combination with the superprompt, three modes of ranking the substitutes were finalised. The first is a similarity-based ranking system, the second a frequency-based one, and finally, within the superprompt setting, a prompt-based unranked setting.

4.6 Evaluation

The evaluation for the LS systems is undertaken using an evaluation script provided by the TSAR-2022 Shared Task, which supplies the scores of the evaluation metrics discussed in the previous chapter Saggion et al. (2023) [8]. In the next chapter, the results generated from this script are examined.

To summarise, a brief explanation of the setup made for the experimentation process is laid out. Steps to be undertaken for substitute generation, substitute selection and substitute ranking as well as the system evaluation are laid out.

[8] https://github.com/LaSTUS-TALN-UPF/TSAR-2022-Shared-Task/blob/main/tsar_eval.py

Chapter 5

Experiments and Results

The chapter below provides an overview of the experimental setup. A few changes from the original experimental setup are mentioned as well. Finally, the findings of the configurations are discussed are revealed and briefly discussed.

5.1 The Models

After devising a strategy for the project's methodology, a few issues cropped up during the experimentation phase. Loading large language models requires a lot of computational resources, including a considerable size of CPU RAM as well as GPU access. Due to this issue, loading the models on a typical local machine proved impossible. To tackle this requirement, all experiments were conducted on Google Colaboratory, a hosted Jupyter Notebook service that is especially useful for machine learning experiments [1]. As such, it was possible to work with the models and build a lexical simplification pipeline. The third model, i.e. GPT-2, encountered more issues.

It was found after multiple attempts that GPT-2 did not perform satisfactorily in the case of generating substitutes for complex words in a zero-shot capacity. Despite entering the prompts along with example data, the model was not able to generate any substitutes and instead provided meaningless continuations of the prompt. This may be attributed to the fact that GPT-2 was not necessarily trained to hold conversations, and thus, one couldn't prompt it to produce the required substitute words. As such, although there is a possibility that it could conduct substitute generation with some finetuning, working on it with a zero- or single-shot approach was not fruitful.

Given these results, it was decided to drop GPT-2 from the models to analyse for the project.

5.1.1 Substitute Selection

As noted in the previous chapter, early attempts to select relevant substitutes and filter out irrelevant bits were undertaken using WordNet and cosine similarity. For the former, I used WordNet synonyms for each substitute to look for overlap between them and the system-generated substitutes. Although the idea was decent in theory, it was context-agnostic in that the WordNet synonyms were (naturally) not able to consider the context of the rest of the source sentence. Further experimentation in this direction

[1] https://colab.google/

would have come under the task of Word Sense Disambiguation (WSD), which is not covered within the scope of the current project. Due to this, the plan was scrapped.

Similar problems cropped up in the case of utilizing cosine similarity. Basic cosine similarity calculates the similarity between two vectors of an inner product space. In this case, the vectors are those of word embeddings projected onto a higher dimensional space. Since this was also context-agnostic, it was also removed as an approach.

SentenceBERT

The model utilized to conduct the selection of substitutes was SentenceBERT, the sentence similarity model developed by Reimers and Gurevych (2019) and made accessible with the Hugging Face inference API. The model functions such that it compares the similarity score (on a scale of 0 to 1) between the source sentence (containing the complex word) and the sentences containing the substitutes (which replace the complex word).

The idea to set a threshold value met with a few complications, as the substitute-incorporated sentences yielded some surprising results: Although it was quite accurate in ranking the substitutes according to relevance during the initial findings, the exact similarity score of the best-ranking substitute varied per instance. For example, in an instance of the trial data, although "concealed" was correctly ranked by the similarity ranker as the first in the case of the complex word "disguised", it still returned a similarity score of 0.9689. In comparison, "mandatory" (the highest-ranked substitute) scored 0.9948 in the case of the complex word "compulsory". Indeed, for a few instances, the highest similarity score (for the highest-ranking substitute) was found to be in the range of 0.8 or 0.7. In dealing with such variations in similarity scores, it was naturally impossible to maintain a set threshold.

As such, it was ultimately decided to drop the threshold approach in favour of a more straightforward "top 10" approach. In the case that more than 10 substitutes were generated, the sentence selector retained only the top 10 words with the highest similarity scores. In the case of using it as a ranker, the same idea was retained.

5.2 The Configurations

For each model (i.e., LLaMA - Alpaca-LoRA and Orca), multiple combinations were chosen to be tested as follows:

- **Performance of the model given the prompt template**: This checks how well the model receives and interacts with each prompt template.
- **Performance of the model given the presence of context (with vs. without the context of the sentence)**: Here, we observe how well the output substitutes are generated based on whether the context was present or absent. If the context is included, the prompt is preceded by "Context: [SENTENCE] \n Given the above sentence". The preceding sentence is courtesy of Aumiller and Gertz (2022)
- **Performance of the model based on the ranking system**: Here, the robustness of the ranking system for the substitutes is tested. The two modes are similarity-only (substitutes are ranked from the highest similarity score to the

lowest) and word frequency (substitutes are first filtered (selected) according to similarity scores and then ranked according to ngram frequency)

Prompt	Context	Ranker
"P1: Provide 10 easier substitutes for the word [complex word]"	No	Similarity-only
		Word Frequency
	Yes	Similarity-only
		Word Frequency
"P2: List 10 easier substitutes for the word [complex word]"	No	Similarity-only
		Word Frequency
	Yes	Similarity-only
		Word Frequency

Table 5.1: Configurations for testing per model

For both generative models, the above combinations together add up to 16 setups. In addition to this, the zero- and single-shot prompts are also tested, as part of the superprompt experiments. In total, this adds up to 20 runs.

5.3 Results

This section provides the findings of the configurations listed above. As mentioned previously, the results have been generated with the help of the evaluation script provided by the Shared Task. The results are listed model by model, and further, prompt by prompt. Variations in models, prompt templates, presence of context, and ranking systems are compared.

5.3.1 Alpaca-LoRA

At a glance, it can be noticed that evaluation scores across prompts and settings for Alpaca-LoRA are highly varied, ranging from extremely low (ACC@1 = 0.16) to much higher (ACC@1 = 0.71) levels. Discussed below are the results of the 10 model runs.

Prompt 1 In the case of Prompt 1, the best configuration is observed to be the frequency-ranked no-context run, with an ACC@1 score of 0.56. It also has the highest Accuracy@k@top1 scores in the whole group. Second to it is the similarity-ranked no-context run, which also has higher MAP@K scores (thus hinting at a better quality of relevant substitutes across the lists). The Potential@k scores do not show significant changes with respect to changes in ranking systems.

The poorer runs for this prompt appear to be those with context: one can see that a considerable drop is seen across all metrics when context is introduced. Despite the context-inclusive similarity run having slightly higher Accuracy@k@top1 and MAP@k scores, it has almost the same Potential@k scores.

Prompt 2 Similar to the previous group, the best scoring runs here appear to be the frequency- and similarity-ranked non-context ones (0.51 and 0.50, respectively). Most of the metrics for these two runs seem to be near-identical. However, both still rank lower than their counterparts from the Prompt 1 runs. To an extent, the Accuracy@k@top1

scores across all runs remain to have a difference in the range of 5-6%, which means that the quality of the top-ranked substitutes is consistent regardless of changes in ranker settings.

Of note, it is observed that context-inclusive scores improve notably when the prompt is switched to Prompt 2, which may imply that the model answers to context a little better with the particular prompt template.

Superprompts Inarguably, the model performs best when provided with the two superprompts: the two runs are the only ones for the model that outperform the upper baselines of the TSAR-2022 Shared Task (TSAR-LSBERT). The zero-shot superprompt yields a fair ACC@1 score of 0.64, while also accounting for an Accuracy@1@top1 score of 0.31, which so far is the highest for a zero-shot setting. On the other hand, Accuracy@1@top3 does not show a considerable increase compared to the other runs (however, it is still higher than the second-best Accuracy@1@top3 by 3%). It is interesting to note that the MAP@K scores show the steepest decline among all zero-shot runs at a 47% decrease from MAP@1 to MAP@10.

The single-shot run is the highest-achieving one for Alpaca-LoRA at 0.71, thus indicating that the model operates best when given very explicit examples/instructions. Accuracy@1@top1 is similar to the zero-shot run at 0.34. The metric rises steadily till Accuracy@3@top1 at 0.52. Also like its zero-shot counterpart, the MAP@K also has an extremely steep decline at 53% from MAP@1 to MAP@10. For both the zero-shot and single-shot superprompts, Potential@K scores do not demonstrate any noticeable increase.

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
no-con sim	0.5281	0.1876	0.3056	0.378	0.3644	0.272	0.1534	0.7747	0.831	0.8659
no-con freq	0.5576	0.2198	0.3109	0.3994	0.3599	0.2645	0.1521	0.8016	0.8552	0.8686
w-con sim	0.268	0.1179	0.1769	0.2064	0.159	0.1112	0.0612	0.3672	0.3914	0.4075
w-con freq	0.1581	0.0509	0.1286	0.1742	0.1085	0.0842	0.0494	0.3243	0.3753	0.4075

Table 5.2: Results of Prompt 1: Alpaca-LoRA

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
no-con sim	0.5013	0.1823	0.2868	0.378	0.3515	0.2639	0.1526	0.7345	0.7801	0.8016
no-con freq	0.5067	0.1823	0.327	0.4262	0.3614	0.2662	0.1549	0.7587	0.7908	0.8042
w-con sim	0.4316	0.1849	0.2815	0.3431	0.2868	0.2094	0.1189	0.6032	0.6514	0.6702
w-con freq	0.3806	0.1394	0.2627	0.3458	0.2619	0.1972	0.1127	0.6005	0.6541	0.6702

Table 5.3: Results of Prompt 2: Alpaca-LoRA

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
0-shot SP	0.6434	0.3163	0.4101	0.4557	0.4229	0.3069	0.1693	0.7587	0.8176	0.8364
1-shot SP	0.7158	0.3431	0.4932	0.5281	0.467	0.3408	0.1857	0.8659	0.8793	0.8847

Table 5.4: Results of the superprompts: Alpaca-LoRA

5.3.2 Orca

Compared to the results for the Alpaca-LoRA runs, the variation range of the scores for Orca seems to be much shorter, with the lowest ACC@1 being 0.40 and the highest being 0.76 (the latter being the highest out of all 20 runs). Highlighted below are the detailed results of the 10 Orca configurations.

Prompt 1 It can be seen that scores for Orca for the first two prompts seem to be in line with those of Alpaca-LoRA: indeed, this is more prominent in the case of the no-context setups. Among the two setups for the first prompt, the similarity-ranked one shows an ACC@1 score 10% higher than the frequency-ranked one at 0.51. It also possesses the highest Accuracy@1@top1 score in the group. Interestingly, these observations are opposite to those of Alpaca-LoRA, where frequency-ranked runs score higher. Slightly higher scores are observed across most metrics, especially in the ones where k is 1 or 3. However, the Potential@k scores are very similar.

A big difference between Orca and Alpaca-LoRA is that for both prompts, scores seem to improve with the addition of context (similarity- and frequency-ranked runs scoring ACC@1 scores of 0.53 and 0.43, respectively). Although Accuracy@1@top1 scores seem to be lower than the no-context similarity run, all metrics demonstrate a slight increase in comparison. MAP@K scores do not show significant changes with the introduction of context. The most interesting observation is that the context-inclusive runs show the highest Potential@10 scores for all 20 runs (0.968 and 0.97 for similarity- and frequency-ranked, respectively).

Prompt 2 The Prompt 2 scores do not demonstrate any profound variation across configurations compared to the first prompt. Indeed, one can also observe that most scores between the two groups show only a difference of about 1-2%. This hints at the assumption that a change in prompts does not directly affect the model by a lot, as opposed to what we see for Alpaca-LoRA. The best-performing runs here are also the context-inclusive runs (similarity at ACC@1 = 0.52 followed by frequency at ACC@1 = 0.43), which are almost identical to their Prompt 1 counterparts. In addition, they have the exact same Potential@10 scores too, which are the highest among all setups (as discussed above). As with Prompt 1, the improvement of MAP@K scores after including context seems to be minor at best.

Superprompts It is clear that the Orca Superprompt runs achieve the highest overall scores among all the configurations. Surprisingly, the zero-shot prompt outperforms the single-shot one by 2% (ACC@1 scores being 0.76 and 0.74, respectively). This is directly in contrast to the Alpaca-LoRA superprompt runs, which seem to improve considerably with the one-shot setting. Similarly, their MAP@K scores also show steep declines: MAP@1 to MAP@10 declines by 54% for the zero-shot setup and by 52% for the single-shot. The Accuracy@k@top1 scores are also the most impressive among all the runs. The zero-shot run boasts an Accuracy@1@top1 of 0.39. For the single-shot run, it is 0.43, which is naturally the highest one. Although the Potential@10 runs are not as impressive as those of context-inclusive ones of Prompts 1 and 2, all Potential@K scores range from 0.88 to 0.95.

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
no-con sim	0.5093	0.2037	0.3083	0.4101	0.3641	0.289	0.1826	0.7694	0.8659	0.9249
no-con freq	0.4048	0.1394	0.268	0.3753	0.3098	0.2611	0.17	0.7238	0.8686	0.9276
w-con sim	0.5308	0.1769	0.319	0.4235	0.386	0.3098	0.199	0.8364	0.9276	0.9678
w-con freq	0.4262	0.1501	0.252	0.3699	0.3218	0.2727	0.1829	0.756	0.9115	0.9705

Table 5.5: Results of Prompt 1: Orca

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
no-con sim	0.5013	0.2037	0.3163	0.4262	0.3656	0.2884	0.1831	0.7721	0.8659	0.9329
no-con freq	0.4048	0.1313	0.2654	0.3646	0.3114	0.2611	0.1705	0.7184	0.8713	0.9329
w-con sim	0.5254	0.1849	0.3243	0.445	0.3833	0.3116	0.1985	0.8391	0.9115	0.9678
w-con freq	0.4343	0.1608	0.2627	0.3672	0.3245	0.2742	0.1829	0.756	0.9142	0.9705

Table 5.6: Results of Prompt 2: Orca

	ACC@1	A@k@top1	A@k@top2	A@k@top3	MAP@3	MAP@5	MAP@10	P@3	P@5	P@10
0-shot SP	0.7613	0.3914	0.5254	0.5871	0.525	0.3918	0.2271	0.9222	0.9436	0.9571
1-shot SP	0.7479	0.4262	0.5415	0.5764	0.512	0.3861	0.2247	0.882	0.9222	0.9436

Table 5.7: Results of the superprompts: Orca

In summary, we find that both models, i.e., Alpaca-LoRA and Orca, perform reasonably in providing substitutes for the given data. While Alpaca-LoRA operates better without context, Orca improves its performance in its presence. Barring a few Alpaca-LoRA, it is noted that most similarity-ranked runs outperform their frequency-ranked counterparts. In addition, it is found in both models that using superprompts yields higher scores compared to using more specialised systems for substitute selection and ranking. In the next chapter, these setups are further analyzed to gain a better insight into their failures.

Chapter 6

Error Analysis

Utilising generative LLMs for downstream tasks can often be challenging in some unexpected ways, especially with respect to their behaviours. Given their use of probabilistic methods to predict words (Kucharavy et al., 2023), their responses can often vary, and optimal (or even consistent) answers may not always be guaranteed. As such, this can cause the primary step of the LS pipeline - substitute generation - to cause unsatisfactory results, thereby impairing the output received from the rest of the system. As such, it is important to identify such model failures and their patterns. The current chapter highlights some errors occurring frequently during the substitute generation phase due to the models.

6.1 Common Types of Errors

The most commonly occurring errors within the outputs are listed as follows:

1. Blank output errors
2. Echoing errors
3. Noise errors
4. Repetitive errors
5. Inaccurate rankings
6. Multi-word hallucinations

6.1.1 Blank output errors

As the name states, in the case of a blank output error, the model is unable to generate any substitutes whatsoever and thus only returns blanks. Sometimes this is observed in the case of the word "input" being the only one within the generated list of substitutes. Other times, when absolutely nothing is generated, the code returns the word "error". In the case of post-parsing, blank output errors can also be caused as a consequence of one of the errors mentioned below (see echoing and noise errors). This kind of error is observed the most in the case of Alpaca-LoRA, in context-inclusive runs where blank outputs result in multiple ACC@k=MAP@k=0 scores, thereby drastically reducing the overall performance scores of these runs. On the other hand, non-outputs seem to be extremely rare in the case of Orca.

6.1.2 Echoing errors

An echoing error occurs when the model simply echoes or repeats the given prompt instead of generating meaningful substitutes. Indeed, at times, the model would also generate the same prompt multiple times. This is also observed most frequently in the case of Alpaca-LoRA, especially when context is introduced to the prompt. However, in most cases, the answer parsing tool ensures that the contents of the prompt do not leak into the list of substitutes. In such cases, the list of substitutes remains empty, which makes this error type merge with blank output errors (thereby contributing to the poor scores in this category).

6.1.3 Noise errors

Oftentimes, despite parsing, it is possible for noise to leak into the list of substitutes. This included gibberish (such as generating non-words such as "wordwordword" as well as undecipherable keywords such as "noqa" (see table 6.1). Although it demonstrates model failure, this is also a fault of the parser tool. Another instance is an example from the Orca outputs, where the substitutes list for the word "encodes" consists of meaningless words such as "use", "replace", "eg", "carets", and "myname". Due to its nature, the frequency ranker is likely to rank up words with the highest frequencies, thus bringing down the scores across metrics such as Accuracy@K@top1 and MAP@K. Consequently, the results also appear to be lower in the case of the frequency-ranked setups.

6.1.4 Repetitive errors

In the case of repetitive errors, the model is unable to generate more than one substitute and thus repeats the same word multiple times. Often, it was the case that only a single word would be generated ten times, and post-parsing, there would only be one substitute in the list of substitutes. Other times, after generating a few substitutes, the model would generate one substitute continuously until the model was cut off by its character limit.

In the table below, the word "unexpectedly" was generated 33 times before being cut off (Alpaca-LoRA. Although not as frequently, Orca generations also show this trend, such as when it generates the word "re-enacting" seven times (although the original complex word is the same).

6.1.5 Inaccurate rankings

Naturally, this is the most common of all the errors generated by the model and the most expected one as well. In this error type, the system's ranking of substitutes is not enough to achieve a satisfactory score. In the example in the table below, the system-generated (and model-ranked) substitutes ('overflowing', 'swollen', 'stuffed', 'full', 'overfilled') are not ranked according to the gold standard ('filled', 'overflowing', 'filled out', 'swelling', 'swell',), thereby resulting in a lower MAP@10 score of 0.1.

This issue is seen prominently in the case of Orca, where the Potential@10 scores are high, but the MAP@k scores show steep declines across setups, thus solidifying the assumption that although the model generates substitutes in most instances, the ranking of the substitutes is not the most optimal.

6.1.6 Multi-word hallucinations

Multi-word hallucinations occur when the model does not generate simpler words as requested, and instead generates multi-word phrases that may or may not be synonymous with the original word. In the example given below, the model (Orca) generated "not allowing to participate and "not including in the debate" for "exclusion", though the context does not mention that the person was part of a debate per se. In another instance, it generated "rude and disrespectful children" and "troublesome teenagers" for the word "monsters" (although, within the context, the age group of the children being referred to as "monsters" is never mentioned).

Unsurprisingly, in a few instances, multiple errors show up together in a single answer. For example, for the word "proportionate", Alpaca-LoRA generated "More than half of" about 25 times, with the phrase being too long, contextually erroneous as well as repeated too many times.

Error Type	Context	Answer
Blank output	...“ T dwarfs ”, for which Gliese 229B is the **prototype**.’	Instruction: Instruction: ### Input:
Echoing	“#12-5 Two lorries were partially crushed by the collapse, which was **triggered**...”	“#12-5 Two lorries were partially crushed by the collapse, which was triggered...” (generated five times)
Noise	“...attempt to hold an **unsanctioned** rally...”	# noqa: E501 # noqa: E501 # noqa: E501
Repetitive	“...acting **suspiciously** lately at Bidston Hill...”	4. Unexplainedly 5. Unexpectedly 6. Unexpectedly 7. Unexpectedly 8. Unexpectedly 9. Unexpectedly
Inaccurate rankings	“...to salvage its balance sheet, **bulging** with losses....”	overflowing’, ’swollen’, ’stuffed’, ’full’, ’overfilled’,
Multi-word hallucinations	“#23-2 Everson argues his **exclusion** violates Title 11 of the U.S. Code of Federal Regulations in that debate hosts must not...”	9. Not allowing to participate, 10. Not including in the debate

Table 6.1: Common error types observed in system-generated examples. Note: words in **bold** represent the context words within the respective instances

The chapter thus provides a brief look into the causes behind the variations in scores across models, and how their failures influence their outputs, and consequently, the systems built on their basis.

Chapter 7

Discussion

The previous chapters have provided a detailed exploration of the experimentation conducted for this project, i.e., the usage of open-source models for the task of lexical simplification. A comprehensive background of the task has been furnished, along with its inspiration (the TSAR-2022 Shared Task). This has been followed by details of the experiments, their results and their subsequent analyses. The current chapter delves into an in-depth discussion of the above-mentioned points as well as any future directions derived from the same.

7.1 Discussion of Results

7.1.1 Key Findings

For the 20 total runs tested, there appeared to be a diverse set of scores for every combination, with the best ACC@1 score being 0.76 (Orca zero-shot superprompt) and the poorest being 0.16 (Alpaca-LoRA frequency-ranked, with context). However, it can be gleaned from the results that multiple runs including both models (Orca and Alpaca) tested during this project yielded promising results.

Alpaca-LoRA (LLaMA)

- **Performance based on the prompt template**: Alpaca-LoRA showed considerably varied scores with respect to changes in prompt templates. Notably, its context-inclusive scores only improved for Prompt 2 and performed poorly in the case of Prompt 1. Introducing the superprompts gave a considerable boost to the score (single-shot more so than zero-shot). This leads us to believe that Alpaca-LoRA performs better in case of certain prompts, and best when given clear examples of what is requested of it.

- **Performance given the presence of context**: The general expectation was that given the introduction of context, the model is better able to process the word sense and relations of a particular complex word. However, Alpaca-LoRA does not perform as expected when given the context, especially in zero-shot settings. The worst performance is seen in the case of Prompt 1 (ACC@1 scores of 0.27 and 0.16), leading to the assumption that if it is not given explicit instructions (w.r.t superprompts), the model is not able to process context properly, and thus does not produce any output (blank output error).

- **Performance based on the ranking system**: It is clear in the case of Alpaca-LoRA that in the case of Prompts 1 and 2, the system performs best with a similarity-based ranker rather than a frequency-based one. Where the similarity ranker prioritises relevance, the frequency ranker prioritises simplicity based on frequency. Based on the current data, it seems that the similarity ranker runs perform better. However, both ranking systems do not perform as well as the innate ranking of the model itself (as instructed by the superprompts). As such, one can believe that Alpaca-LoRA performs best when it ranks itself.

Orca

- **Performance based on the prompt template**: In the case of the first two prompts, results by Orca are extremely similar. As such, any differences between the metrics seem too minor (ranging broadly between 0.5% and 1%) to remark upon. As such, it can be interpreted that, unlike Alpaca-LoRA, Orca can handle variations in prompts adequately (though further experimentation in prompting would be required to consolidate this claim). Orca also seems to improve its scores, especially in the case of generating and ranking top substitutes, with the help of the superprompts. This is seen clearly in the case of the single-shot run, which means that its performance improves with example data.

- **Performance given the presence of context**: Across both prompt settings, it is noticeable that Orca shows slight improvement with the addition of context in the case of ACC@1. Interestingly, however, the addition also causes a drop in Accuracy1k@top1 scores in many setups (barring frequency-ranked runs). However, this is offset by the increase in MAP@k scores with context, which means that although top candidates may not always match, the number of correctly ranked relevant candidates still shows a slight improvement.

- **Performance based on the ranking system**: Orca largely follows the same trends as Alpaca-LoRA with respect to ranking systems. Across all setups for Prompts 1 and 2, similarity-ranked systems outperform frequency-ranked ones, and this holds true for almost all metrics. Where there isn't a slight improvement in the case of similarity rankers, metrics appear to be near-equal. Also similarly to its counterpart, Orca performs best when it ranks itself, showing a considerable increase across most of the metrics. The two models thus consolidate the observation that instead of being aided by sophisticated selection and ranking systems, generative models perform best when they rank the substitutes themselves.

7.1.2 Notable Achievements

Out of 20, 18 of the runs outperformed the lower baseline (TSAR-TUNER) (on the basis of ACC@1). 16 runs also had higher Accuracy@1@top scores. Further, 4 runs also outperformed the upper baseline (LSBERT). The best-performing run is also nearly on par with the second run of UniHD (ACC@1 = 0.77), and Orca's single-shot superprompt run also scored an Accuracy@1@top (0.4262) rivalling that of UniHD's top run (0.4289) (Saggion et al., 2023). From these figures, it can be assumed that both Alpaca and Orca are able to compete with the performance of state-of-the-art systems for the task of lexical simplification despite having modest parameter counts (7 billion and 13 billion,

respectively). This observation also reinforces the assertion of Hoffmann et al. (2022) that small models trained on more data yield promising performances.

7.2 Significance

The public release of GPT-3 has led to a novel public interest in the operation of generative models in the last few years, especially in the case of its application in downstream tasks (Zong and Krishnamachari, 2022). Originally, UniHD's entry for the TSAR-2022 Shared Task opened the avenue for conducting lexical simplification with the use of generative large language models. The idea that merely *prompting* the model to provide substitutes for a given complex word was tested by Aumiller and Gertz (2022) and Vásquez-Rodríguez et al. (2022) and yielded promising results. Further, the nature of generative models and their responses to a variety of prompts has led to the development of interest in the recently established study of prompt engineering.

Consequently, the primary motivation behind this project was derived. The aim to maintain accessibility and transparency in research, especially in the field of natural language processing, has become more relevant than ever. Thus, I decided to evaluate the performance of open-source generative models in the substitute generation phase of the LS pipeline.

While the results of the selected models did not exactly outperform the best-performing system (i.e., GPT-3), they still delivered promising results for all three major aspects of lexical simplification, i.e., substitute generation, selection and ranking all by themselves. Currently, the usage of vLLMs (very large language models) carries various ecological implications [1]. The models provide considerable competition to the winner of the Shared Task despite having modest capacities, which means that there is a possibility of mitigating computational constraints in the future for similar experiments and tasks.

The findings of this project lead us to believe that generative models can be implemented to perform the task of lexical simplification. Further, in doing so, they are able to achieve significantly promising results (calculated with respect to the Shared Task). This has significant implications for the field, as it means that these models, open-source and free-to-access, have the potential to further streamline complex NLP tasks. As more and more open-source models are developed, the issues that originally arose about using closed-source models can be rectified, thereby democratising the accessibility of generative large language models.

7.3 Limitations

7.3.1 Computational Requirements

A notable limitation regarding this project is that the systems, which require multiple models to enable the pipeline, come with significant computational costs. Unsurprisingly, large language models, consisting of billions of parameters, naturally require considerable computational power to load and operate. This often includes a certain amount of system RAM as well as GPU resources that are not easy to come by on a local machine. Although free resources such as virtual machines exist, these come with

[1] https://www.wwt.com/article/the-impact-of-generative-ai-and-large-language-models-on-organizational-sustainability-and-esg-goals

their own rate limits that may hinder continuous experimentation. Another component was the similarity API hosted by Hugging Face, which may impose its own rate limits.

7.3.2 Model Sizes

Although it is mentioned above that the models used for the project require considerable computational power, it should be noted that the version of LLaMA utilised consisted only of 7B parameters, which means that its comparison with Orca does not remain fully fair. In the future, this could be handled by choosing models having the same capacities.

7.3.3 Lack of Morphological Adaptation

Oftentimes, lexical simplification pipelines require one final component before the list of substitutes is finalised. This is considered to be the morphological generation and contextual adaptation of the substitutes. Given the fact that English is not a highly inflected language as well as the assumption that the models could handle such nuance, MA was not included in the project. However, it would not be amiss to have this component if a similar system were created in another language.

7.3.4 English-specific Experimentation

The system (and consequently, the pipeline) only takes into consideration the English language. Various components depend upon the same, such as the models (which are not explicitly mentioned to be multilingual). as well as the above-mentioned lack of the MA component. As such, multiple changes would need to be implemented in order to make the system compatible with other languages (especially linguistically diverse and inflected ones).

7.3.5 Reproducibility

As discussed previously, the nature of generative models does not guarantee that the results produced would always be the same, even when the prompts and configurations are the same. Hence, such experimentation cannot account for absolute reproducibility.

7.4 Future Work

7.4.1 Exploring Different Model Settings

The behaviour of generative models can differ with varying configuration settings such as temperature, top_p and more. These settings control whether the response will be conservative/safe or creative/diverse. The scope of the current project did not cover multiple model settings, but in the future, it would be interesting to test multiple settings to see how they influence substitute generation.

7.4.2 Leveraging Different Models

Currently, there seems to be an abundance of new generative models being released frequently. The current project only focused on two, but leveraging multiple state-

of-the-art models emerging at the moment will definitely prove fruitful in terms of variations and improvements in results.

7.4.3 Language-specific and Multilingual Models

Similarly, it would be beneficial to expand the scope of LS systems so that multiple languages are covered. Of new language models being launched right now, many of these may be language-specific or multilingual, and testing such models will help expand the coverage of LS systems across multiple languages. In turn, this would have the possibility to help provide for the reading and accessibility requirements of speakers of those languages.

7.4.4 Few-shot and Ensemble Prompts

Although the current project listed a total of 20 setups for the two models evaluated, the only two prompt techniques utilised were zero- and single-shot. It was established during experimentation that generative models learnt best with the help of example data. In the same vein, if the models are given more examples, i.e., through few-shot learning, it would possibly enable the model to learn the data better. In doing so, it would be able to improve its performance.

The UniHD entry for the Shared Task utilised an "ensemble" technique, wherein model outputs for each one of six prompts were combined to make a comprehensive list of substitutes. This was done in order to smooth out any hallucinations generated by one or more of the prompt responses, with the filtering process finally leaving a comprehensive list of substitutes. It was this ensemble run that won the Shared Task, and as such, replicating this technique is highly likely to improve the system's performance.

In summary, the chapter provides a detailed analysis of the results and their implications, along with explanations regarding particular scores. Some scores worthy of introspection are discussed, along with their comparison to the baselines and winners of the Shared Task. The impact of the research in the field is also briefly discussed, and finally, the project's limitations and directions for future work are also highlighted.

Chapter 8

Conclusion

While investigating the usefulness of building a lexical simplification system using open-source models, this research has made a few noteworthy contributions:

1. It is confirmed that open-source generative models, specifically Orca and Alpaca, can successfully be utilised for prompt learning as tools to generate simplified text, offering a viable approach to addressing the lexical simplification challenge.

2. A comparative analysis between Orca and Alpaca presents the strengths and limitations of each model. Notably, Orca's adeptness at context understanding and Alpaca's contextual struggles highlight areas for further investigation and enhancement.

3. The behaviour of Orca and Alpaca with respect to external rankers in comparison to their own ranking criteria may suggest an interesting direction for future research.

4. It is shown that open-source models exhibit the potential to achieve near-state-of-the-art performance in lexical simplification tasks.

The findings shed light on the answers to the research questions asked at the start of this study:

Primary Research Question: Can one successfully leverage open-source models for prompt learning to build a lexical simplification system for the English language?

The answer to the primary research question can certainly be derived from the findings garnered by the study. Yes, one can successfully leverage open-source models for prompt learning to build a lexical simplification system for the English language. Indeed, prompt learning is a highly efficient technique that is uncomplicated and still able to yield decent results.

Sub-question 1: How does the performance of these generative models compare against each other?

Despite their differences in capacities, both models did not have drastic differences in performance in the case of their best respective runs. However, from the findings, we note that Orca (13B params) outperforms Alpaca-LoRA (Llama 7B params) but not by a considerable degree. Orca takes well to context, while Alpaca does not. Both

models do not perform as well with external rankers (such as similarity and frequency-based) as they do when they rank themselves, hinting at an innate understanding of the substitute ranking process.

Sub-question 2: How does their performance compare against that of the current state-of-the-art lexical simplification systems?

The results of the majority of the total runs were competitive in the context of the TSAR-2022 Shared Task on Lexical Simplification, successfully outperforming the lower baseline (LS-TUNER, the non-neural state-of-the-art baseline). While the best runs (i.e., the superprompt runs) of both models yielded promising results, they scored slightly lower than the winner of the TSAR-2022 task i.e. UniHD. However, both of them surpass the upper baseline of the task i.e. LSBERT. Orca's is almost the same as the second run of UniHD (ranked 2nd in the task). This speaks for the capability of open-source models in performing almost as well as closed-source/proprietary state-of-the-art generative models.

In conclusion, the current research in the fields of text simplification and generative large language models answers critical questions posed about the capabilities of the latter. Indeed, the research also improves upon our understanding of the potential of these models for prompt learning in building lexical simplification systems. After conducting a substantial number of analyses and comparisons, the project offers a considerable contribution to these fields. Consequently, this can also inspire the direction for continued exploration and innovation in the same.

Bibliography

S. S. Alqahtani. Technology-based interventions for children with reading difficulties: A literature review from 2010 to 2020 - educational technology research and development, Nov 2020. URL https://link.springer.com/article/10.1007/s11423-020-09859-1.

S. Aluísio and C. Gasperin. Fostering digital inclusion and accessibility: The PorSimples project for simplification of Portuguese texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53, Los Angeles, California, June 2010. Association for Computational Linguistics. URL https://aclanthology.org/W10-1607.

D. Aumiller and M. Gertz. UniHD at TSAR-2022 shared task: Is compute all we need for lexical simplification? In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 251–258, Abu Dhabi, United Arab Emirates (Virtual), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.tsar-1.28.

O. Biran, S. Brody, and N. Elhadad. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-2087.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

D. Ferrés, H. Saggion, and X. Gómez Guinovart. An adaptable lexical simplification architecture for major Ibero-Romance languages. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 40–47, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5406. URL https://aclanthology.org/W17-5406.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022.

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.

D. Jurafsky and J. H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson, 2022.

T. Kajiwara, H. Matsumoto, and K. Yamamoto. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*, pages 59–73, Kaohsiung, Taiwan, Oct. 2013. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP). URL https://aclanthology.org/O13-1007.

D. Kauchak and R. Barzilay. Paraphrasing for automatic evaluation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 455–462, New York City, USA, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/N06-1058.

A. Kucharavy, Z. Schillaci, L. Maréchal, M. Würsch, L. Dolamic, R. Sabonnadiere, D. P. David, A. Mermoud, and V. Lenders. Fundamentals of generative large language models and perspectives in cyber-defense, 2023.

A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc, O. Stanley, R. Nagyfi, S. ES, S. Suri, D. Glushkov, A. Dantuluri, A. Maguire, C. Schuhmann, H. Nguyen, and A. Mattick. Openassistant conversations – democratizing large language model alignment, 2023.

J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, T. G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. L. Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, 2011. doi: 10.1126/science.1199644. URL https://www.science.org/doi/abs/10.1126/science.1199644.

G. Miller. *WordNet*. MIT Press, 1998.

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

G. H. Paetzold and L. Specia. A survey on lexical simplification. *J. Artif. Int. Res.*, 60 (1):549–593, sep 2017a. ISSN 1076-9757.

G. H. Paetzold and L. Specia. A survey on lexical simplification, 2017b. URL https://doi.org/10.1613/jair.5526.

J. Qiang, Y. Li, Y. Zhu, Y. Yuan, Y. Shi, and X. Wu. Lsbert: Lexical simplification based on bert. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3064–3076, sep 2021. ISSN 2329-9290. doi: 10.1109/TASLP.2021.3111589. URL https://doi.org/10.1109/TASLP.2021.3111589.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL http://arxiv.org/abs/1908.10084.

H. Saggion, S. Štajner, D. Ferrés, K. C. Sheang, M. Shardlow, K. North, and M. Zampieri. Findings of the tsar-2022 shared task on multilingual lexical simplification, 2023.

M. Shardlow. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1), 2014. URL `http://dx.doi.org/10.14569/SpecialIssue.2014.040109`.

S. Stajner, D. Ferres, M. Shardlow, K. North, M. Zampieri, and H. Saggion. Lexical simplification benchmarks for english, portuguese, and spanish, 2022.

S. R. Thomas and S. Anderson. Wordnet-based lexical simplification of a document. 09 2012.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

L. Vásquez-Rodríguez, N. T. H. Nguyen, M. Shardlow, and S. Ananiadou. Uom&mmu at tsar-2022 shared task: Prompt learning for lexical simplification. In *Shared Task on Text Simplification, Accessibility, and Readability (TSAR-2022), EMNLP 2022*, Dec. 2022.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023.

S. M. Yimam, C. Biemann, S. Malmasi, G. Paetzold, L. Specia, S. Štajner, A. Tack, and M. Zampieri. A report on the complex word identification shared task 2018. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-0507. URL `https://aclanthology.org/W18-0507`.

M. Zong and B. Krishnamachari. a survey on gpt-3, 2022.

S. Štajner, D. Ferrés, M. Shardlow, K. North, M. Zampieri, and H. Saggion. Lexical simplification benchmarks for english, portuguese, and spanish. *Frontiers in Artificial Intelligence*, 5, 2022. ISSN 2624-8212. doi: 10.3389/frai.2022.991242. URL `https://www.frontiersin.org/articles/10.3389/frai.2022.991242`.