Unsupervised Syntactic Text Simplification with AMR

by

Kostyantyn Guzhva

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Syntactic text simplification, the task of reducing the grammatical complexity of text while preserving the content, can be useful for non-native speakers, text summarization, and other downstream natural language processing tasks. Many traditional methods are rule-based and do not generalize, while methods that rely on modern large language models often are limited by prohibitive computational requirements or data privacy concerns. We present a text simplification pipeline based on Abstract Meaning Representation which can run on modest hardware, and report on intrinsic and extrinsic evaluations of its performance. We find that our method achieves comparable performance to GPT-3.5, at a fraction of the cost, and without any privacy concerns. Additionally, it outperforms a best in class rule based text simplifier. To see if our simplified text preserves the semantics of the original text, we evaluate our simplified text in two downstream tasks: relation extraction, and entity linking. We find that our syntactic simplification pipeline has limited or no impact on the performance of the methods we evaluate for these tasks, indicating that our pipeline preserves the information in the original text.

Preface

This work was completed in collaboration with Peiran Yao, a PhD student at the University of Alberta working with Denilson Barbosa. In particular, Peiran worked on speeding up the finetuning process, wrote the code for the intrinsic evaluation and ran it, as well as implemented the sub-graph selection algorithm. The ideas behind the method were discussed and implemented collaboratively. All other experiments, analysis, and writing were done independently.

Acknowledgements

Thanks to my supervisor Denilson Barbosa for his guidance, patience, and support. Thanks to Peiran Yao for his collaboration on this work. Thanks to Scotiabank for the funding they provided. Thanks to Carrie Demmans Epp and Alona Fyshe, the exam committee members, as well as Greg Kondrak, the chair of the exam, for their help and feedback.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

Syntactic text simplification is the task of reducing the grammatical complexity of text while preserving the content [14], [34]. In practice, this often amounts to replacing long multi-clause sentences with multiple, simpler and shorter sentences that convey the same information as the original text. There is a complementary task, text simplification, which involves both syntactic simplification and *lexical* simplification – replacing different words with synonyms that are easier to understand [35]. Text summarization is another form of simplification which also involves removing superfluous information or unnecessary details [2].

We focus specifically on syntactic simplification in this work. The objective is to simplify complex multi-clause sentences by splitting them into multiple syntactically simpler sentences with no dependent clauses. These syntactically simpler sentences should have the same information as the original sentence. Specifically, we aim to preserve the semantics of the original sentence. As an example of syntactic simplification, consider the following sentence:

> In 1935 they moved back to Chaldon, to live in a primitive stone cottage known as 24 West Chaldon.

Our method simplifies the sentence into the following three sentences:

- They lived in primitive stone cottages.

- They moved back to Chaldon to live in 1935.
- The primitive stone cottages were known as 24 West Chaldon.

The method we propose relies on Abstract Meaning Representation (AMR) as an intermediate hierarchical representation which we use to select sub-sentences. All of the sentences in the example are extracted from the AMR graph of the original sentence, which can be seen in Figure 1.1. We are not overly concerned about the fluency of the sub-sentences we extract (how well they read), and we do not consider the order of the sentences. These things are out of scope, and they are not captured well by the metrics we use to evaluate our system. One practical consequence of this is that the sub-sentences we extract may be ordered in a way that makes the simplified text less clear. In the example above, the sub-sentences are not necessarily in the same order as the ideas from the original sentence, but in this case it doesn't seriously impact the meaning. Additionally, due to limitations in AMR, which we will discuss later, it may not always be possible to preserve all of the semantics of the original sentence.

1.2 Motivation

Syntactic text simplification is important because it can help humans understand complicated language, and it has also been used as a pre-processing step for other downstream NLP tasks.

Syntactic text simplification is a part of the broader task of text simplification, which includes lexical simplification. Text simplification can be useful for language learners, non-native speakers, or those with low literacy levels trying to understand text [14], [31]. It can also be helpful to those with disabilities; Niklaus *et al.* [35] cite works describing its utility to people with aphasia, dyslexia, or deafness.

In the past text simplification was used for a multitude of downstream NLP tasks. Other works which do text simplification (syntactic, lexical, or both) cite the following use cases: summarization, parsing, semantic role labeling, in-

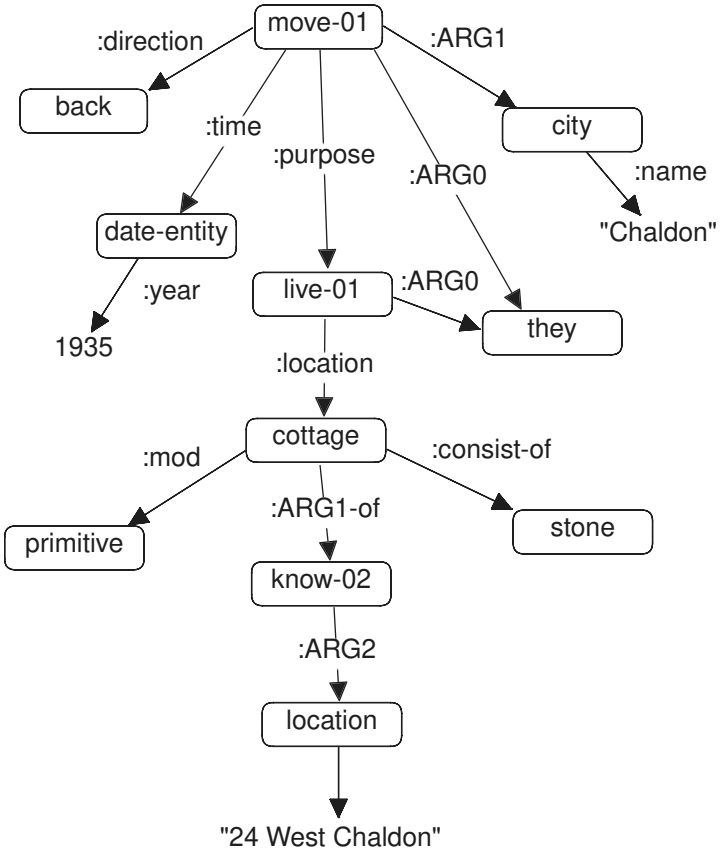

Figure 1.1: AMR graph for the sentence "In 1935 they moved back to Chaldon, to live in a primitive stone cottage known as 24 West Chaldon."

formation extraction, and machine translation [14], [31], [32], [36]. Admittedly, these papers cite works which show text simplification to be useful mostly for more traditional NLP methods, which are unlike modern methods that use large models as end-to-end solutions. In this thesis we will examine whether our syntactic simplification is useful as a preprocessing step for two relatively modern methods on two different NLP tasks.

1.3 Overview of Thesis Work

There are many previous approaches to text simplification [2], and they can broadly be categorized into methods which are: supervised [30], [59], unsupervised [50], [54], those which depend on some kind of parse tree [34], [49], [54], and those which use handwritten rules [34]. Note that these categories are not mutually exclusive. Supervised neural methods often frame text simplification as a translation task where gold labelled data is required. Both the creation of such a dataset and model training can be costly. Rule-based methods can be effective, but may suffer from a lack of generality and may require a lot of human effort to develop good rules.

We introduce an unsupervised syntactic text simplification method based on AMR, which does not suffer from these problems. Our approach does not require gold data, and we only do some finetuning of a pre-trained model. We only have one algorithm for generating simplified sentences from an AMR representation, but our method depends on having a good AMR parser and AMR-to-text model.

One reason why syntactic text simplification (and the broader task of text simplification) is challenging is that it is hard to formally describe how the simplified text should look in general, and therefore it is hard to evaluate the quality of the simplifications. In Section 2.5 we will discuss how there is no consensus about which metrics should be used to evaluate proposed text simplification implementations. There are many papers which are critical of commonly used metrics, and those which propose new metrics. Given the diversity of tasks and approaches related to text simplification, it is difficult

to rely on a single metric to evaluate the quality of the simplified text. We follow the practice reported in recent papers of using multiple metrics for our intrinsic evaluation [14], [30], [50], [54].

1.4 Research Questions

We aim to address the following research questions:

- How can we build a syntactic simplification system using AMR?
- Can such a system be competitive with modern large language models?
- How useful is our syntactically simpler text in downstream natural language processing tasks?

Regarding the first question: there are other approaches to text simplification which use intermediate semantic representations [21], [31], [32], [49], so it is certainly possible to build text simplification systems using semantic representations in general. We believe that the structure of AMR makes extracting syntactically simpler sentences very convenient, and we present a simple and intuitive algorithm for doing so in Section 4.4.

The idea behind the second research question is to investigate whether it is worth the effort to build systems like ours, when there are large language models (LLMs) available that can already simplify text. Recent LLMs have demonstrated the ability to achieve state of the art performance in many natural language processing (NLP) tasks as zero-shot methods [9]. Feng *et al.* [17] show that modern LLMs already outperform previous text simplification methods. We show our method is comparable to the performance of ChatGPT-3.5 for the task of syntactic sentence simplification. Our method is outperformed by some metrics, but we will discuss some other merits of our approach versus LLMs.

For the third research question, we want to know if our simplified text is still useful for modern NLP systems, since syntactic simplification was used as a pre-processing step for NLP tasks in the past. Additionally, we would like

our system to preserve the meaning of the sentences it simplifies. Whether or not a system preserves meaning is difficult to capture with standard metrics (though in Section 2.5 we describe how some metrics, like BERTScore, might do this) so we test our method on two downstream NLP tasks to see if the same information can be extracted from the original text can be extracted from our simplified text. We show that our method does not really improve the performance of the systems we test on the downstream NLP tasks, but does not necessarily degrade performance either, suggesting it preserves information in the original text.

1.5 Outline

In Chapter 2 we go into more detail about AMR, as well as the metrics we use for evaluation. We also discuss the transformer architecture and BERT specifically, which is the foundation of many methods we use in our extrinsic evaluation. Finally, we describe AMRBART [4], the model we use for AMR parsing and AMR-to-text generation.

Chapter 3 introduces other related methods which do syntactic simplification. Like we have mentioned before, the methods are broadly neural methods (supervised or unsupervised) or they make use of hand-written rules. A number of the methods use some type of intermediate hierarchical representation, like a dependency parse. We also discuss methods similar to ours which use different semantic representations.

In Chapter 4 we fully describe our syntactic simplification pipeline, including the role of AMR parsing, how we use AMRBART for our purposes, and how to extract sub-sentences from AMR.

We perform an intrinsic and extrinsic evaluation in Chapter 5. During the intrinsic evaluation we compare our approach against two other methods, ChatGPT-3.5 and DisSim [34] on two standard datasets. In terms of SARI, likely the most relevant metric, we achieve better performance than DisSim, and comparable performance to ChatGPT-3.5. For the extrinsic evaluation we use our simplified text for two downstream NLP tasks: entity linking, and

relation extraction. By achieving comparable results to the original text in these experiments, we show that our simplified text does not lose very much information.

Overall, our results indicate that our text simplification method is competitive. Unlike ChatGPT, it can be hosted locally and without the data privacy concerns of depending on a third party API. The model can be run on fairly modest hardware, and the cost of electricity to do inference is much cheaper than ChatGPT API access.

Chapter 2

Background

2.1 Syntactic Text Simplification

Syntactic text simplification is the task of reducing the grammatical complexity of text while preserving the original meaning. Syntactic text simplification is a type of text simplification, which is the broader problem of reducing the linguistic complexity of text while preserving the original meaning. Al-Thanyyan and Azmi [2] categorize text simplification into two types: lexical simplification, and syntactic simplification. Lexical simplification is the process of replacing complex words with simpler synonyms, whereas syntactic simplification is the process of reducing the grammatical complexity of text. A related problem is text summarization, which tries to reduce the length of the text while preserving the important information. Summarization can also involve both lexical and syntactic simplification.

In the past, text simplification was a pre-processing step for some NLP tasks, such as parsing, question generation, information extraction, fact retrieval, semantic role labelling, machine translation, relation extraction, and summarization [2], [49]. Text simplification might still be useful for some NLP tasks today, but it seems less likely with the prevalence of large language models (LLMs) which can be finetuned to provide an end-to-end solution for many NLP tasks. Even if the utility of text simplifiction as a pre-processing step is limited, syntactic text simplification can still help humans understand grammatically complex text. In this case, computationally expensive LLMs may not be necessary to generate high quality syntactically simpler sentences.

In this thesis, we focus specifically on syntactic text simplification. Our approach involves simplifying the grammatical complexity by splitting a sentence into multiple shorter and less grammatically complex ones.

At this point, we need to address some potential ambiguity with the terminology used in the literature: what is really syntactic simplification is sometimes called sentence splitting in the relevant literature, with slight changes to the task definition. For example, Narayan *et al.* [33] introduce the "Split and Rephrase" task, which they describe as splitting a complex sentence into a meaning-preserving sequence of shorter sentences. The paper distinguishes split and rephrase from simplification by saying that simplification is not necessarily meaning preserving, and that deletions are allowed in split and rephrase. Other work simply introduces sentence splitting as a type of syntactic simplification [25]. Surya *et al.* [50] describe text simplification in general as consisting of three major types of operations: splitting, deletion/compression, and paraphrasing. Lee and Don [25] note the existence of previous works dating back to 1996 which also do sentence splitting, but acknowledge it as a form of syntactic simplification. Niklaus *et al.* [34] present a rule-based approach to syntactic simplification, and describe the problem as we do.

We argue that sentence splitting is simply a type of syntactic simplification because it is one way to reduce the grammatical complexity of text. This position is supported by previous literature which does not attempt to redefine the task. Finally, it is worth mentioning that there are ways of doing sentence-level syntactic simplification without splitting sentences, for example by shortening sentences or deleting clauses [50]. However, we will show in the related work that most approaches involve sentence splitting.

2.2 Abstract Meaning Representation

Abstract meaning representation (AMR), first introduced by Langkilde and Knight [24], is a language representation that aims to capture the semantics of text. For the purposes of this thesis, AMR can be thought of as a sentence representation similar to a dependency parse or a constituency parse. Unlike

a dependency parse or a constituency parse, an AMR representation does not necessarily depend on the precise grammatical structure of a sentence. Two syntactically different sentences could have a similar AMR representation if they have similar semantics. AMR has gained popularity since Banarescu *et al.* [5] proposed using it for statistical language understanding, with the idea that it would encourage work in the field like Penn Treebank did for statistical parsing. Indeed, as Bai *et al.* [4] points out, AMR has been shown to be useful for NLP tasks such as text summarization, machine translation, information extraction, and dialogue systems.

Banarescu *et al.* [5] describe AMRs as rooted, directed, edge-labeled, leaf-labeled graphs. AMR graphs are represented in a format called PENMAN notation [6]. In PENMAN, each node has an associated variable, a concept, and potentially multiple arguments or attribute relations associated with the concept. A variable is like a name, which can be used to refer to a node elsewhere in the graph. A concept is either a word from the sentence being represented, a PropBank frameset [39], or a special keyword. In PropBank, an annotated corpus of semantic roles, a frameset refers to a distinct usage of a verb in a particular role, which is a way of using the verb. A triple consisting of a concept, its associated argument, and the subject of the argument (which can be either another variable or a literal) forms a relation. Figure 2.1 shows the PENMAN representation for the AMR parse of a sentence, and Figure 2.2 the corresponding graph representation. The first node, `move-01` is a concept. It has both arguments (`ARG0`, `ARG1`) and attributes (`direction`, `time`, `purpose`). Some nodes have attributes that point directly to literals, like the triple (`city`, `name`, `"Chaldon"`).

2.3 BERT and the Transformer

BERT, or Bidirectional Encoder Representations from Transformers, is a language model based on the transformer architecture [15], [53]. Shortly after its release, BERT became ubiquitous in NLP. Many researchers practiced BERTology, where they just applied BERT to various problems in NLP (often

```
(z0 / move-01
  :ARG0 (z1 / they)
  :ARG1 (z2 / city
    :name (z3 / name
      :op1 "Chaldon"))
  :direction (z4 / back)
  :time (z5 / date-entity
    :year 1935)
  :purpose (z6 / live-01
    :ARG0 z1
    :location (z7 / cottage
      :consist-of (z8 / stone)
      :mod (z9 / primitive)
      :ARG1-of (z10 / know-02
        :ARG2 (z11 / location
          :name (z12 / name
            :op1 24
            :op2 "West"
            :op3 "Chaldon"))))))
```

Figure 2.1: PENMAN representation of the AMR parse for the sentence "In 1935 they moved back to Chaldon, to live in a primitive stone cottage known as 24 West Chaldon."

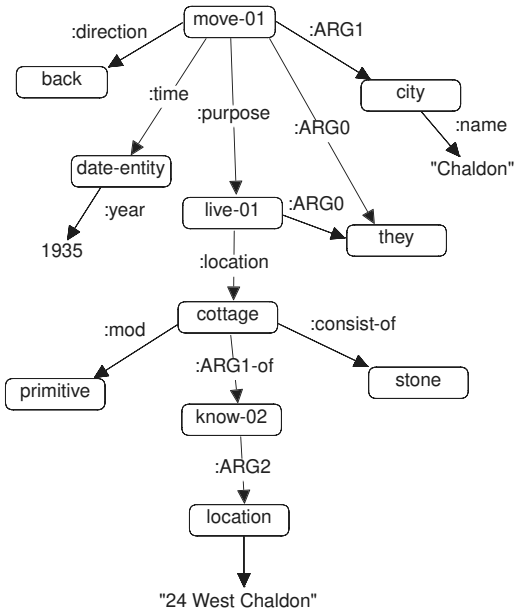

Figure 2.2: AMR graph for the sentence "In 1935 they moved back to Chaldon, to live in a primitive stone cottage known as 24 West Chaldon." This figure is a repeat of Figure 1.1 for convenience.

with state of the art results) [42]. Suffice to say, BERT and the transformer architecture are very important. In this thesis we use some methods which are based on BERT in our experiments, so in this section we briefly discuss BERT and the transformer.

Vaswani *et al.* [53] propose the transformer as a replacement to recurrent models, which used to be very popular for NLP. Recurrent models were state of the art in many NLP tasks prior to the transformer. However, recurrent models had their problems: their computation was difficult to parallelize due to their sequential nature [53], learning long-range dependencies was a challenge [53], they often experienced deteriorated performance on old tasks when acquiring new knowledge (known as catastrophic forgetting) [13], and they suffered from the vanishing or exploding gradients problem [41]. The main innovation of the transformer is that it completely replaces recurrence with the attention function, which Vaswani *et al.* [53] describe as mapping a query and a set of key-value pairs to an output, where the query, key-value pairs, and output are all vectors. Intuitively, attention adds trainable parameters which weight the importance of parts of the input to the output. The transformer described in Vaswani *et al.* [53] has an encoder-decoder architecture, following previous successes in NLP [12]. The encoder component of the architecture *encodes* a variable length input into a fixed length vector representation, and the decoder attempts to *decode* the vector to recover the original sequence.

The main innovation of BERT is that it is just the bidirectional encoder (the B in BERT). The architecture proposed by Vaswani *et al.* [53] has a unidirectional transformer decoder, meaning the model only considers outputs at positions less than i when making a prediction for position i in a sequence. BERT uses a "masked language model" pre-training objective where some tokens from the input are randomly masked, and the objective is to predict the masked token based on the context to its left and right, making BERT a bidirectional transformer. Otherwise, BERT's architecture is so similar to the original transformer that the authors of BERT simply refer the reader to Vaswani *et al.* [53]. Devlin *et al.* [15] empirically demonstrate the effectiveness of their architecture by fine-tuning BERT to obtain state of the art results on

11 NLP tasks.

2.4 AMRBART

AMRBART is a model capable of AMR parsing and AMR-to-text generation [4]. AMR parsing is the task of transforming a sentence into its corresponding AMR graph, and AMR-to-text is the opposite task of generating text from an AMR graph. The syntactic sentence simplification method described in this thesis relies on AMRBART for both tasks.

AMRBART is based on the Seq2Seq model BART [28], [51], which is a denoising autoencoder based on the transformer architecture. BART is like BERT, but it also has a decoder, making it more suitable for text generation. BART's pre-training objective is similar to BERT's, but instead of just masking input tokens, BART is trained to reconstruct "corrupted" input sequences. The corruption involves token masking, but also token deletion, text infilling (multiple tokens replaced by one token), sentence permutation (shuffling sentences), and document rotation (shifting tokens). These transformations make the model better at generalizing. BART is particularly effective at text generation, which is useful here since AMRBART treats AMR parsing and AMR-to-text as Seq2Seq text generation tasks.

AMRBART treats the AMR-to-text and AMR parsing problems as Seq2Seq problems by linearizing AMR graphs into a format not so different than PENMAN, but with special tokens to handle co-referring nodes and to mark the beginning and end of graphs. Bai *et al.* [4] pre-train BART on these linearized AMR graphs, before pre-training further with something they call the unified pre-training framework. The pre-training method combines the text sequences and AMR sequences passed as input to the model, and the objective is to reconstruct the corrupted sequence of graph or text, depending on whether we are doing AMR-to-text or text-to-AMR. Bai *et al.* [4] say that this way the model can benefit from training on both the text and the AMR by enforcing learning the correspondence between the two. Pre-training and finetuning also share the same input format, but the finetuning will have an empty text or

graph – Bai *et al.* [4] claim this helps to facilitate knowledge transfer between pre-training and finetuning.

As far as we know, AMRBART still achieves the best performance on a number of datasets for which the authors report results in both AMR parsing and AMR-to-text. AMRBART is slightly outperformed by other methods on a few datasets for the AMR parsing task; those methods do not do AMR-to-text [23], [26]. In the AMR-to-text task, one method has slightly better performance on some datasets, but not others [11]. Given that no other method convincingly beats AMRBART across multiple datasets in neither AMR parsing nor AMR-to-text generation, we believe it is a good choice for both tasks.

2.5 Metrics

In this thesis we use several different metrics to evaluate syntactic simplification methods. This is because there is no consensus in the literature about what metric is the most appropriate, and there are multiple papers criticizing existing commonly used metrics [48], [52], [57]. The metrics we consider are: bilingual evaluation understudy (BLEU) [40], SARI [57], BERTScore [58], and Flesch-Kincaid Grade Level (FKGL) [22]. We also discuss SAMSA [48] and why we do not use it. Of the metrics discussed here, BLEU and SARI are the most commonly reported, with SARI being the preferred metric because it was specifically developed for text simplification. Nonetheless, the other metrics are still seen in the literature.

BLEU is an extremely popular metric for evaluating machine translation tasks, but it is also a popular choice for text simplification. BLEU measures the similarity between a candidate text (such as a translation, or in this thesis a simplified sentence) and potentially many reference text. BLEU is calculated as:

$$\text{BLEU} = \text{BP} \times \exp(\sum_{n=1}^{N} w_n \log(p_n))$$

where N is the number of reference strings, $n \in \{1, ..., N\}$, p_n is the modified n-gram precision, w_n are typically uniform weights, and BP is a brevity

penalty term which can be set to penalize overly short strings.

The modified n-gram precision is simply the fraction of n-grams from the candidate text which appear in the reference texts. Since this definition of the modified n-gram precision gives an excessively high score to the shortest string which contains all of the reference n-grams, the brevity penalty is often set to penalize short candidate texts. BLEU is always between 0 and 1, but a candidate text will not achieve a BLEU score of 1 unless it is identical (in terms of n-grams) to the reference text.

Since BLEU is based on n-gram precision relative to the reference text, it has some limitations. First of all, BLEU is highly dependent on having good references to compare to. Secondly, BLEU does not account for lexical substitutions or paraphrasing, causing semantically similar texts to receive a lower score than they should. These are not necessarily major obstacles for syntactic simplification, though other work has shown that BLEU might not be the most appropriate metric for text simplification. Sulem *et al.* [47] shows that BLEU can negatively correlate with human judgements of simplicity, arguing that it is a particularly bad metric for sentence splitting. Instead, the authors advocate for their own metric, SAMSA [48]. Xu *et al.* [57] show that although BLEU correlates strongly with meaning and grammaticality, it does not correlate well with simplicity.

SARI is a metric which compares System output Against References and against the Input sentence. Like BLEU, SARI is computed from n-grams, however SARI is designed specifically for evaluating sentence simplification. Xu *et al.* [57] describe it as a metric which explicitly measures the goodness of words that are added, deleted, and kept by the systems. The formula for SARI is:

$$\text{SARI} = \frac{F_{add} + F_{keep} + P_{del}}{3}$$

with

$$P_{operation} = \frac{1}{k}\sum_{n=1}^{k} p_{operation}(n)$$

$$R_{operation} = \frac{1}{k}\sum_{n=1}^{k} r_{operation}(n)$$

$$F_{operation} = \frac{2 \times P_{operation} \times R_{operation}}{P_{operation} + R_{operation}}$$

where k is the highest n-gram order, and $p_{operation}(n)$ and $r_{operation}(n)$ are a special precision and recall between the candidate text and the reference text. They are defined differently for each of the three operations (add, keep, delete) – for more details, see the paper Xu *et al.* [57]. The authors show that SARI is highly correlated with human judgements of simplicity, and that BLEU does not correlate well with simplicity. Based on their findings, the authors argue that BLEU is insufficient for evaluating text simplification.

BERTScore is an automatic evaluation metric for text generation based on contextual embeddings from the language representation model BERT [15], [58]. Unlike BLEU and SARI, which just consider exact n-gram precision or recall, BERTScore computes the similarity as a sum of cosine similarities between the token embeddings of the sentences being compared. One advantage of this approach is that BERTScore can capture paraphrases. The authors show that BERTScore has a higher correlation to human judgements on various tasks compared to BLEU and some other metrics. The authors also claim their metric is task agnostic, which should make it suitable for text simplification (though they do not specifically evaluate text simplification).

BERTScore can be calculated from a reference sentence $x = \langle x_1, ..., x_n \rangle$ and a candidate sentence $\hat{x} = \langle \hat{x_1}, ..., \hat{x_m} \rangle$. For the calculation, tokens are represented as contextual embeddings generated by BERT. With pre-normalized vectors, the cosine similarity of a reference token x_i and candidate token $\hat{x_j}$ can be computed as $\mathbf{x}_i^\top \hat{\mathbf{x}}_j$. The full BERTScore calculation matches each token in x to a token in $\hat{x}$ to compute recall, and it also matches each token in $\hat{x}$ to a token in x to compute precision, and tokens are matched greedily. The formula for recall is:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x_j} \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j$$

and the formula for precision is:

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x_j} \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j$$

In this thesis we report BERTScore F1, which is computed from the precision and recall as:

$$F_{\text{BERT}} = 2\frac{R_{\text{BERT}} \times P_{\text{BERT}}}{R_{\text{BERT}} + P_{\text{BERT}}}$$

BLEURT [44] is another BERT-based metric similar to BERTScore which is used to compare a candidate sentence and reference sentence. Unlike BERTScore, BLEURT passes both the candidate and reference sentences to a single model which computes a score indicative of how well the candidate sentence relates to the reference sentence. BLEURT is not commonly used in literature for text simplification evaluations, so we do not report it.

FKGL is a simple formula to measure the readability of text. It is based on the total number of words, the number of syllables in the words, and the total numbers of sentences in a text. This is the formula:

$$\text{FKGL} = 0.39(\frac{\text{total words}}{\text{total sentences}}) + 11.8(\frac{\text{total syllables}}{\text{total words}}) - 15.59$$

It is also easy to interpret; an FKGL of 8 indicates that an 8th grade U.S. student can easily read the text. Although FKGL has been used to evaluate text simplification systems, Tanprasert and Kauchak [52] argue that it should not be used based on the findings that very basic post-processing can drastically improve FKGL while having minimal impact on BLEU and SARI. The analysis is compelling, so we use other metrics in this thesis, but we keep FKGL anyway like the previous literature in the field.

SAMSA is an acronym for Simplification Automatic evaluation Measure through Semantic Annotation. It is another text simplification metric which

aims to address the structural aspects of the task [48]. SAMSA is based on the Universal Cognitive Conceptual Annotation (UCCA) [1]. UCCA is a semantic annotation scheme similar to AMR. The main reason why we do not use SAMSA is because we believe it is overly tuned for a simplification method based on UCCA by the same authors as SAMSA [49]. In particular, SAMSA penalizes simplifications (by assigning them a score of zero) which have more output sentences than the number of scenes (semantic units) in the UCCA parse.

Chapter 3

Related Work

The literature on text simplification is vast, and dates back decades [2]. To keep the literature review relevant, we focus only on relatively recent methods which have an emphasis on syntactic text simplification. Among those methods, we cannot fully compare against all of them due to slightly different problem definitions; for example, a number of the methods we discuss also do lexical simplification, but they are still relevant due to their contributions to syntactic simplification.

Broadly speaking, contemporary approaches are either rule-based or neural methods. The neural methods can be classified as either supervised or unsupervised. The rule-based methods sometimes take advantage of neural networks somewhere in the process, like to generate an intermediate representation or parse the text, but ultimately the syntactic simplification comes from handwritten rules. Both methods often take advantage of some intermediate representation, like a dependency parse. We separately discuss syntactic simplification methods based on semantic representations like ours.

3.1 Rule based Methods

Rule based methods for text simplification are those which rely on a number of rules, usually hand-written by humans, to simplify text. Rule based methods were some of the earliest ways of doing syntactic simplification. Some disadvantages are that they typically require labor from humans to create the rules, validating these rules for correctness can be difficult, and the rules likely

will not generalize to other languages. Despite this, rule-based methods could achieve state of the art performance at least until until Niklaus *et al.* [35].

Early methods used rules written for chunked text (e.g. words) and parts-of-speech tags, but later methods usually used hierarchical representations of text like constituency or dependency parses [10], [46]. The method by Glavaš and Štajner [20] is a relatively recent example of syntactic simplification which still uses rules over text, though they also make use of some dependency relations. Siddharthan and Mandya [46] present a rule-based method over dependency parses which uses 111 handcrafted rules for sentence splitting, and they also automatically derive thousands of rules for lexical simplifications and deletion operations. Most notably, there is DisSim [35] which is a fairly recent rule-based method with 35 rules that achieved state of the art performance at the time of publication.

3.2 Neural Methods

Neural methods use a neural network to simplify text. After the publication of Sutskever *et al.* [51], there were supervised methods which treated text simplification as a machine translation task and just used a Seq2Seq model as-is [38]. Due to the way these methods were trained and implemented, they did full text simplification (lexical and syntactic). Zhang and Lapata [59] outperformed competitive methods at the time by augmenting an encoder-decoder architecture with reinforcement learning. Their method does syntactic simplification, though they also present a version which can do lexical simplification. Martin *et al.* [30] also use a supervised Seq2Seq model, and they introduce a way to parameterize the model to control several aspects of the simplified text, like the amount of compression, paraphrasing, and the lexical and syntactic complexity.

Surya *et al.* [50] present the earliest unsupervised system we could find that outperforms previous supervised text simplification systems at the time, and they do lexical simplification as well. Some recent unsupervised neural methods make use of dependency trees to tackle the task of syntactic simplification

[14], [54].

We would be remiss not to mention large language models (LLMs), which have shown strong or even state of the art performance in many NLP tasks [9]. Feng *et al.* [17] perform an analysis of the performance of ChatGPT and GPT-3.5 compared to more traditional supervised and unsupervised neural methods on the sentence simplification task. Sentence simplification is a broader problem than syntactic simplification, as it includes lexical simplification, but syntactic simplification is still involved. They find that not only are previously state of the art methods outperformed by LLMs on standard metrics like SARI and FKGL, but also humans prefer the LLM outputs. The human evaluation is based on subjective measures of simplicity, adequacy, fluency, as well as rank relative to other samples, and is performed by 3 non-native English speakers described as having medium language ability.

3.3 Methods based on Semantic Representations

The earliest syntactic simplification method based on a semantic representation that we were able to find is Narayan and Gardent [31]. The method is based on a type of representation called Discourse Representation Structure (DRS), and sentences are simplified according to a combination of several pretrained probabilistic models. The same authors later develop an unsupervised approach based on the same semantic representation [32].

Sulem *et al.* [49] present a text simplification method based on the semantic representation UCCA [1]. It is a hybrid approach, which does rule-based sentence splitting based on the UCCA representation, the output of which is passed to a neural component for further simplification.

Finally, there is GRASS (GRAph-based Semantic represenation for syntactic Simplification) [21] which is a rule-based method built on top of yet another semantic representation.

Chapter 4

Method

In this chapter we describe how we use AMR and AMRBART to do unsupervised syntactic sentence simplification.

At a high level, our method parses sentences into AMR graphs, extracts sub-sentences from the AMR graphs, and then converts them back into text. To better preserve entities, we replace mentions to entities with special tokens when the sub-graphs are selected, and place the entities back in the text after the AMR-to-text step. This entire process is described by Figure 4.1.

The method we present is unsupervised; we do not explicitly train our model to learn any kind of correspondence between unsimplified and simplified text. We do, however, finetune AMRBART to improve the quality of its translations.

In a preliminary qualitative evaluation of AMRBART, we noticed that the AMR-to-text step produced strange artifacts in the output text. After parsing a sentence to AMR and then running the text-to-AMR, the output would sometimes have HTML tags or special characters which were not in the original text. This is likely due to the data that was used to train the model; AMRBART is trained on the LDC2017T10 and LDC2020T02 datasets, which were created manually and should not have any HTML or special characters. However, BART, the base for AMRBART, is trained on a combination of news, books, stories, and web text [28]. It is likely that the web text was not fully sanitized. For an example of this, see Table 4.2. Additionally, AMRBART would sometimes drop entities (proper nouns) or replace them with entirely

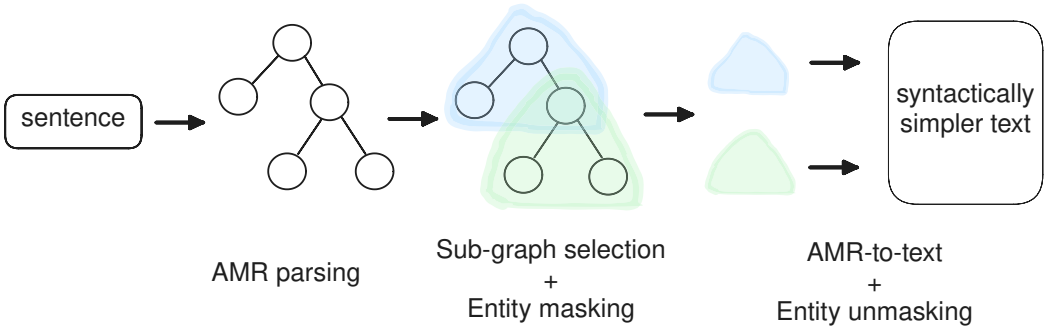

Figure 4.1: Syntactic simplification pipeline. The entity masking happens after the sub-graph selection, and the entities are unmasked after the AMR-to-text model is run on the sub-graphs. For more details on this, see Section 4.3.

different entities. To address these issues, we finetune the model and introduce a masking step.

Finally, we describe our algorithm for extracting multiple syntactically simpler sentences from the AMR parse of a complex sentence, and the full syntactic simplification pipeline using finetuned AMRBART.

4.1 AMR Parsing

We use AMRBART as-is for AMR parsing. It works well, but sometimes the AMR graph it generates is invalid. Recall from Chapter 2 that AMR is represented in PENMAN notation. When AMRBART does AMR parsing, it is supposed to produce a valid AMR graph in PENMAN notation. What we mean when we say that the AMR is invalid is that the PENMAN produced by AMRBART cannot be read into a graph data structure unambiguously because of problems with the PENMAN representation. Since AMRBART is a neural method, it is not guaranteed to always generate a syntactically correct AMR graph. In two specific cases there is an easy fix, which we apply during the pipeline:

1. There is a cycle in the graph. AMR graphs with cycles are invalid. To fix this, we simply break the last-seen edge which causes the cycle.

2. There is a reference to a non-existent node. In these cases we just remove

the reference.

In other cases it is difficult to determine what is wrong with the graph. When AMRBART cannot generate a valid AMR graph for a sentence, we simply output the un-simplified sentence as-is. The reason we do not pass the malformed AMR graph to the AMR-to-text model (which will likely produce okay text) is that our simplification pipeline is based on the AMR graph, as we will describe in this section. Of course, outputting un-simplified sentences can lower the performance of our pipeline according to our metrics, but AMRBART fails to parse sentences extremely infrequently. On the Simplewiki dataset of 1,900,948 sentences, which we describe in the next section, AMRBART fails to parse only 592 sentences or about 0.03%.

4.2 Finetuning AMRBART

To address the problem of strange artifacts in the output resulting from running AMR parsing on the text and then running the AMR-to-text model on the AMR graphs, we finetune AMRBART on a subset of Simple English Wikipedia (Simplewiki) using a dump from Jan 1, 2023. More recent dumps are available on the Wikimedia website[1]. The idea behind using Simplewiki sentences for finetuning is that the AMR graphs of the sentences should already be relatively small because the sentences are short and simple, and therefore the graphs will be representative of the AMR graphs we will be passing to the AMR-to-text model. Our sample of Simplewiki has an FKGL of only 6.38, which means that the text should be easily readable by a U.S. sixth grader, and suggests that the sentences are short and the words have few syllables.

From the Simplewiki dump, we extract approximately 1.9 million sentences and create train/validation/test splits of size 1,894,356/3,000/3,000. Since we do not observe any major issues with the AMR parser, we use it to create training data for the finetuning. This way we can still prepare the model for syntactic simplification in an unsupervised way. We use AMRBART to

[1] `https://dumps.wikimedia.org/`

Original sentence	AMRBART sentence
Stoford is a village in the county of Somerset, England.	Stofordis a village in Somerset, England.
In 546 BCE, Achaemenid Persians was in control of Ankara (approximated date).	Ankara(Armenian: Ankara) was controlled by the Iranian Armenians in approximately 546 BC.
Kidman was then in more movies like Flirting, Billy Bathgate, Far and Away, Malice, My Life and To Die For.	Then Kidman was in more movies like Flirting (f3000) , Billy Bathgate (f3001) , Far and Away (f3002) , Malice (f3003) , My Life (f3004) and To Die For (f3005)
The Serengeti lion: a study of predator–prey relations.	Serengeti lions - Study of predator- prey relation
In music, E is a note, sometimes referred to as "Mi".	<a href="">is a musical note sometimes referred to as Mi.

Table 4.1: Selected outputs from AMRBART on sentences from the Simplewiki dataset which demonstrate how AMRBART (without finetuning) sometimes inserts HTML tags or unusual characters. The original sentences are run though the AMR parser and then the output through AMR-to-text. To save space, the full URLs in the hrefs are omitted. The URLs were Wikipedia links, and Simplewiki dataset also comes from Wikipedia, but we observed this behavior in other datasets as well.

generate the AMR graphs, and we use the original sentences as the gold data for AMR-to-text.

We use the training scripts (with their default hyper-parameters) provided in the AMRBART Github repository[2] to finetune the model for 30 epochs on the training data. The task is to generate the original Simplewiki sentence given the AMR graph for that sentence. After finetuning, the finetuned model achieves a BLEU of 46.23 on the test set, whereas the base model has a BLEU of 39.53. In addition to the improved BLEU, we do not observe HTML or special characters when manually checking the output.

[2] `https://github.com/goodbai-nlp/AMRBART`

4.3 Entity Masking

We handle the issue of AMRBART dropping entities or replacing them with entirely different entities by introducing a masking step. The entity masking happens outside of the model. After AMRBART generates the AMR graph for a sentence, we can identify any entities in the AMR graph by checking for nodes in the AMR graph which have the `name` attribute. For each AMR graph, we replace the name of each entity with a special numbered `ENTITYN` token, where `N` is the $\texttt{N}^{\text{th}}$ entity occurring in the graph. Figure 4.4 shows how the two entity names `"Chaldon"` and `"24 West Chaldon"` are replaced with `"ENTITY1"` and `"ENTITY2"` during the masking step. After the AMR-to-text step we replace occurrences of each `ENTITYN` with the original name of the entity. AMRBART had to be finetuned again to keep the `ENTITYN` tokens in the text during the AMR-to-text step. The finetuned model with masking achieves a BLEU of 59.36 on the Simplewiki data, higher than the finetuned model with no masking. This model was again finetuned for 30 epochs which was enough for the BLEU and loss to begin to plateau; see Figure 4.2 and Figure 4.3.

On the Simplewiki test data, the masked model keeps 5,386 entities from the original text (based on exact matches) compared to the 4,776 entities kept by the base model – a nearly 13% improvement.

Alhough our approach of entity masking by replacing text directly from the inputs and outputs rather than using a special token in the model might seem a little unprincipled, empirically it works very well and it is easy to implement. The reason why our approach works well for AMRBART but perhaps might not for more traditional probabilistic language models comes down to the tokenizer. AMRBART, being based on BART, uses a Byte-Pair Encoding (BPE) tokenizer. BPE was originally proposed as a text compression algorithm [18], but Sennrich *et al.* [45] demonstrate its utility as a simple and effective tokenizer which can help address the problem of out-of-vocabulary (OOV) words. BPE has been used in many popular transformer models like GPT, GPT-2, and of course BART. To create a character-level BPE, start

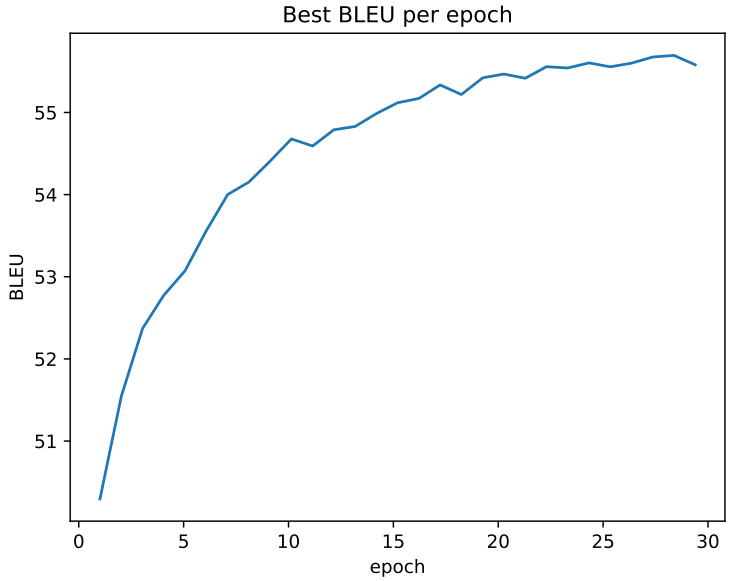

Figure 4.2: Best BLEU per epoch of finetuning AMRBART on AMR-to-text with masking. BLEU begins to plateau towards epoch 30. Note the y-axis does not start at 0.

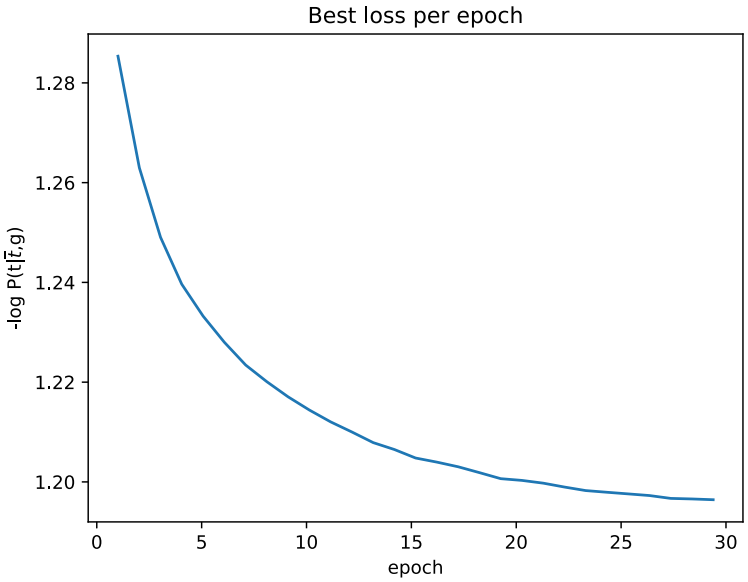

Figure 4.3: Best loss per epoch of finetuning AMRBART on AMR-to-text with masking. Loss begins to plateau towards epoch 30. The loss function is $-\log P(t|\bar{t}, g)$, as defined in Bai *et al.* [4]. t is the text, and g is the AMR graph. In the finetuning setting, $\bar{t}$ is an empty string. Note the y-axis does not start at 0.

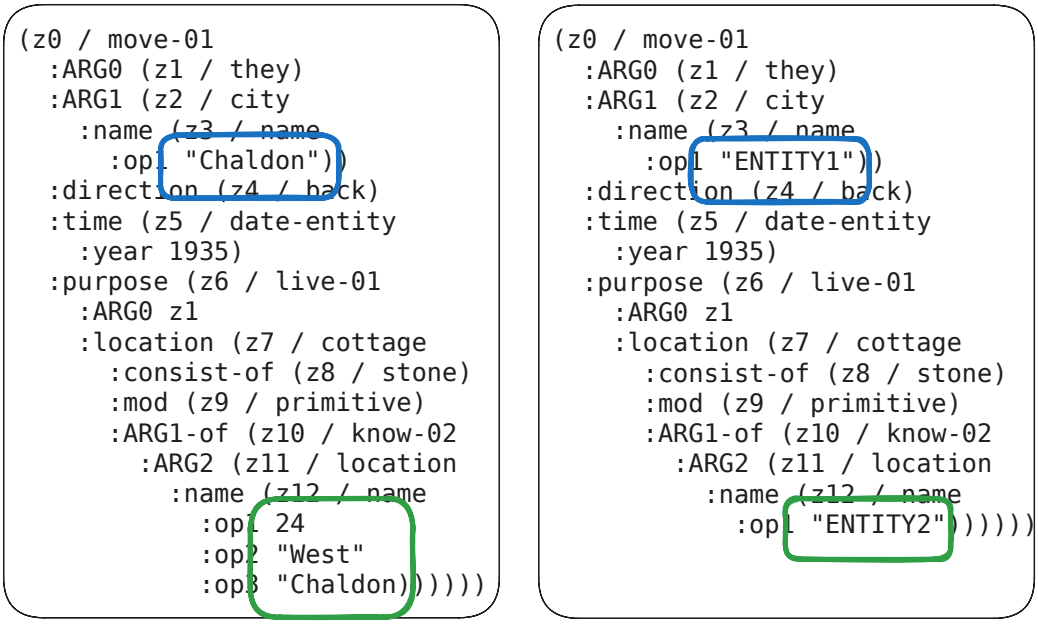

Figure 4.4: Example of how entity masking looks on the PENMAN representation. The entity "Chaldon" is replaced with "ENTITY1" and the entity "24 West Chaldon", which is represented as multiple attributes, is replaced with "ENTITY2".

with the vocabulary of characters and iteratively merge the most frequent pairs from a text until a vocabulary of the desired size is reached. Common n-grams resulting from the merge operation may also be merged. This creates a final vocabulary of all of the starting characters, plus all subword units resulting from the merge operations. With this encoding there is no possible way to have OOV words as long as the character vocabulary is comprehensive.

What this means for our entity masking approach is that AMRBART can just learn to copy instances of `ENTITYN` from the input to the output rather than relying on other information from the AMR graph to copy various different subword units representing arbitrary entity names. The BPE used by AMRBART probably represents the `ENTITYN` tokens as several subword units, which is not ideal. It may be better to have each possible `ENTITYN` as a seperate token in the vocabulary.

4.4 Sub-graph Selection Algorithm

Our method does syntactic simplification by splitting a sentence into smaller and less complex sentences. We do the sentence splitting by generating an AMR graph for a sentence and then splitting the AMR graph into smaller sub-graphs which later become separate sentences.

The sub-graph selection works by first finding all of the nodes corresponding to verbs in the AMR graph. This information is available through the AMR parse. Each verb becomes the parent of its own sub-graph. The arguments of the verb are added to the sub-graph, along with any attributes of those arguments. If any sub-graph (or node) has less than 3 nodes after this procedure and is not yet part of any other sub-graph, it is added to its largest parent sub-graph instead of becoming a separate sentence.

Figure 4.4 illustrates this process. The verbs in the AMR graph in the example are `move-01`, `live-01`, and `know-02`. We create a sub-graph for each verb and all `ARG` edges are added to the graph, along with any attributes like `time`, `direction`, or `purpose` in the case of the verb `move-01`. Note that the `know-02` verb is actually the parent of its graph because `cottage` is an argument of `know-02` through the edge label `ARG1-of`. The last step is to add the attributes of any arguments to the sub-graph. For example, `move-01` has the argument `city` with attribute `name` which needs to be added to the graph.

4.5 Syntactic Simplification Pipeline

Finally we can describe the full end-to-end pipeline for simplifying a document. Although our method simplifies text at the sentence level, we can simplify an entire document by splitting it into sentences, simplifying those sentences, and then combining them.

The complete pipeline to simplify a document follows these steps, also shown in Figure 4.1:

1. Segment the document into sentences using pySBD[3] [43]

[3] `https://github.com/nipunsadvilkar/pySBD`

2. Parse the sentences into AMR graphs (left side Figure 4.5)

3. Mask the entities

4. Select sub-graphs according to the sub-graph selection algorithm

5. Translate sub-graphs into text using AMRBART (right side Figure 4.5)

6. Unmask the entities

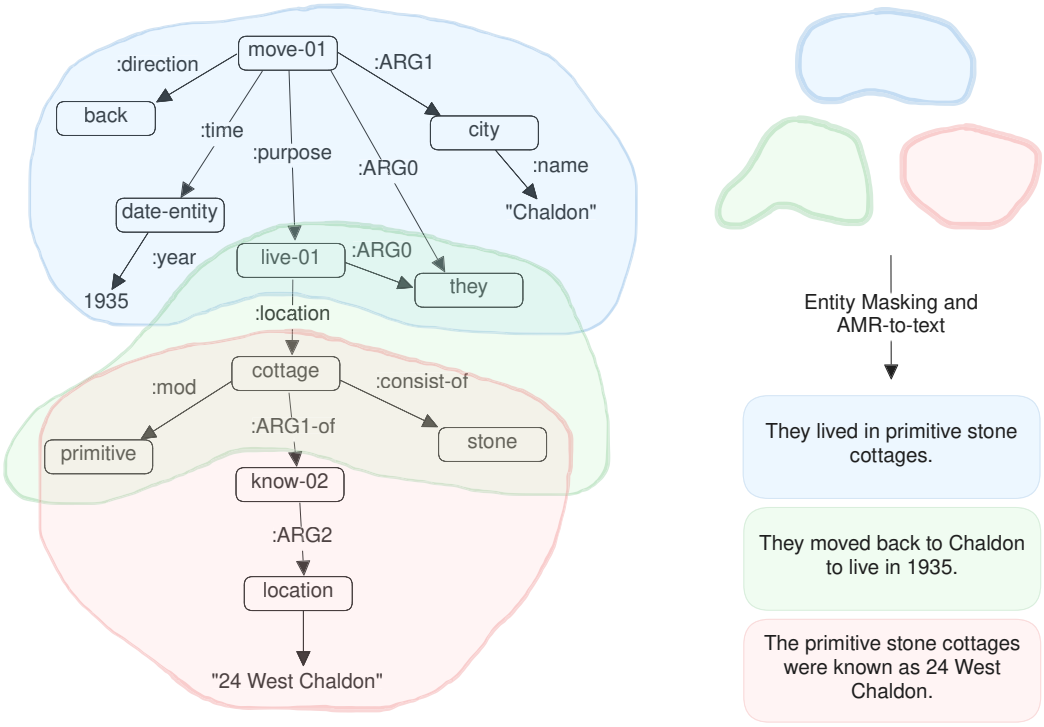

Figure 4.5: Visualization of the sub-graph selection algorithm. The AMR parsing and sub-graph extraction is depicted on the left. After the sentence is parsed into an AMR graph, sub-graphs are extracted from the AMR parse based on the verbs. Entity masking is done before running AMR-to-text on each sub-graph. AMR-to-text is run on each sub-graph to get the new sentences.

Chapter 5

Results

We conduct intrinsic and extrinsic evaluations to demonstrate the effectiveness of our method. The purpose of the intrinsic evaluation is to compare performance against other strong simplification methods using metrics discussed in Chapter 2. Our intention with the extrinsic evaluation is demonstrate that our syntactic simplification pipeline does not lose very much information by using it for downstream NLP tasks, in the sense that the performance of downstream NLP tasks run on the simplified text is not significantly degraded compared to the original text.

5.1 Intrinsic Evaluation

5.1.1 Experimental Setting

We benchmark our syntactic simplification method against ChatGPT-3.5[1], and DisSim [34]. DisSim is the strongest rule-based syntactic simplification method that we are aware of, and at the time of publication it outperformed state of the art supervised and unsupervised neural methods [34]. Since then superior neural methods have been developed, but Feng *et al.* [17] show that LLMs outperform current state of the art sentence simplification methods according to both standard automatic evaluation metrics as well as human judgements, making ChatGPT-3.5 a very strong benchmark.

When we evaluate ChatGPT-3.5, we use the *gpt-3.5-turbo-0301* model with the following system prompt: "You are a helpful assistant that simplifies syn-

[1] `https://openai.com/blog/chatgpt`

tactic structures. You do not use any clauses or conjunctions." and the following user prompt: "Rewrite the following paragraph using simple sentence structures and no clauses or conjunctions:", followed by the text. We arrived at this prompt by following some common prompt engineering techniques. We manually examined the output resulting from this prompt to ensure that ChatGPT-3.5 was simplifying text in the way described.

We use two common datasets for evaluating sentence simplification: WebSplit [33], and MinWikiSplit [37]. MinWikiSplit was created by running DisSim on the WikiSplit dataset [7], a dataset created from Wikipedia edit history. MinWikiSplit consists of sentences broken down into minimal propositions, and their corresponding original sentence. WebSplit is derived from the WebNLG dataset [19], a dataset which maps relational triples (from a knowledge base called DBPedia) to text. Table 5.1.1 shows some examples from the MinWikiSplit dataset. The WebSplit dataset looks similar, but it has more reference sentences with different variations in the form of re-phrasings or different ways of splitting the original sentence. This is useful for the purpose of computing reference-based metrics like BLEU.

For each dataset we report the following standard metrics which we discussed in the background section: BLEU, SARI, BERTScore F1, and FKGL. We report the corpus-level results for each metric, in accordance with other work [34], [54], [57]. We use the Easier Automatic Sentence Simplification Evaluation (EASSE) Python library for computing all of the standard metrics [3]. In addition to the standard metrics, we report the compression ratio and the number of splits. The compression ratio is the length of the simplified text divided by the length of the original text. The number of splits is how many sentences are in the simplified text. For all of our experiments we report the results for our pipeline with AMRBART finetuned (f), masked (m), both (mf), and neither.

Note that compression ratio does not necessarily measure syntactic complexity; a higher or lower compression is not necessarily indicative of text which has a higher or lower syntactic complexity. We include compression ratio as a metric for those curious about how much more text we produce to

Original sentence	Reference sentences
It eventually developed into the Music House Museum and opened the entire display in May 1984.	It eventually developed into the Music House Museum. It eventually opened the entire display in May 1984.
Gandhi lives in Mumbai with his wife and two children, and runs the Mahatma Gandhi Foundation.	Gandhi lives in Mumbai with his wife. Gandhi lives in Mumbai with two children. Gandhi runs the Mahatma Gandhi Foundation.
After attending local schools, he went to the Royal Naval College, Greenwich, and in 1942 joined the Fleet Air Arm.	He went to the Royal Naval College. He in 1942 joined the Fleet Air Arm. He was attending local schools. The Royal Naval College was Greenwich.
Leber was sentenced to death, and executed on 5 January 1945 at Plötzensee Prison in Berlin.	Leber was sentenced to death. Leber was executed on 5 January 1945 at Plötzensee Prison in Berlin.

Table 5.1: Some shorter examples from the MinWikiSplit dataset. The original sentences are on the left, and on the right are the reference sentences used for computing metrics.

simplify a sentence. In fact, we expect the compression ratio for our method to be greater than 1, because splitting one sentence into multiple may involve repeating other words, like connectives or modifiers.

5.1.2 Experiments

The results of the evaluation on WebSplit are in Table 5.2. Our pipeline with finetuned AMRBART has higher SARI than DisSim, and has a comparable BERTScore, but worse BLEU and FKGL. All of our models generate less text than DisSim, indicated by the lower compression ratio. Both DisSim and our method is outperformed by ChatGPT-3.5 on this dataset across all metrics.

The results of the evaluation on MinWikiSplit are in Table 5.3. Our pipeline using AMRBART with finetuning and masking achieves high SARI, only slightly lower than ChatGPT-3.5, and outperforms ChatGPT-3.5 in BLEU, but generates more text and has lower BERTScore.

Model	BLEU	SARI	FKGL	BERTScore	Compression Ratio	Splits
AMR pipeline (mf)	49.54	46.09	6.83	0.47	1.04	1.36
AMR pipeline (m)	29.07	42.69	7.01	0.52	1.07	1.34
AMR pipeline (f)	26.82	46.51	7.29	0.48	1.14	1.38
AMR pipeline	46.55	45.59	6.08	0.49	1.13	1.39
ChatGPT-3.5	77.37	50.53	4.19	0.61	1.05	2.15
DisSim	58.85	42.91	4.75	0.50	1.20	2.33
Exact Copy	50.48	20.16	8.52	0.54	1.00	1.00
Gold	100.00	72.85	5.32	1.00	1.32	1.00

Table 5.2: Intrinsic evaluation results for WebSplit corpus.

Model	BLEU	SARI	FKGL	BERTScore	Compression Ratio	Splits
AMR pipeline (mf)	47.83	36.96	7.50	0.59	1.12	2.18
AMR pipeline (m)	37.46	32.73	7.12	0.50	1.11	2.17
AMR pipeline (f)	53.46	36.66	7.66	0.51	1.19	2.22
AMR pipeline	46.63	35.20	7.08	0.51	1.14	2.17
ChatGPT-3.5	47.11	38.06	5.28	0.68	0.98	2.52
DisSim	-	-	-	-	-	-
Exact Copy	79.16	28.06	13.27	0.79	1.00	1.00
Gold	100.00	100.00	5.15	1.00	1.26	3.49

Table 5.3: Intrinsic evaluation results for MinWikiSplit corpus. The gold data is from DisSim, which is why we omit the metrics for DisSim.

5.1.3 Discussion

Overall, our approach leads to high SARI scores which are better than DisSim. ChatGPT-3.5 achieves better SARI than our model on both WebSplit and MinWikiSplit, but our SARI is comparable on the MinWikiSplit dataset. We generate less text than DisSim with a comparable BERTScore on the WebSplit dataset. Our method does not reduce FKGL as much as DisSim and ChatGPT-3.5. As mentioned earlier, FKGL is calculated from syllable count, word count, and sentence count. Since we do not do any deliberate lexical simplification, we mainly just see improvements in FKGL from shorter sentences. However, it is possible that the AMR-to-text model does some unintentional lexical simplification.

As mentioned earlier, the metrics we compute are corpus-level (in line with the literature). For metrics like BLEU and SARI, this means computing n-gram statistics using candidate and reference text pairs, and then computing the metric for the entire corpus using the overall n-gram statistics. This is different from computing BLEU or SARI for each pair of candidate and reference sentence and averaging them, which would give shorter sentences a

disproportionate contribution to the final metrics relative to their length.

5.1.4 Cost Analysis

In terms of hardware, all inference is done on a single NVIDIA A40 GPU. Our method takes 30 minutes on an NVIDIA A40 GPU to fully process approximately 500k characters. This cost about $0.02 USD in electricty at the time of the experiment. These 500k characters correspond approximately to 128k tokens, and with the output from ChatGPT being similar in length to the input, ChatGPT has to process roughly 256k tokens to simplify the same amount of data. Using the ChatGPT-3.5 pricing[2] at the time of the experiment for the number of input/output tokens required, the cost of using ChatGPT-3.5 is approximately $0.45 USD, making our model many times more cost effective for comparable performance on the task of syntactic simplification.

5.2 Extrinsic Evaluation

To demonstrate that our simplification method preserves the information in the original text, we perform an extrinsic evaluation on two separate tasks: relation extraction, and entity linking.

5.2.1 Relation Extraction

For our first extrinsic evaluation task we evaluate how our syntactic sentence simplification pipeline effects the performance of a relation extraction system. Relation extraction is the task of determining whether text expresses a relationship between entities in the real world, and if so, the relation expressed. For this experiment we use the Orlando dataset [8], which we chose for its high syntactic complexity in the hopes that our simplification pipeline would improve the performance of the relation extraction system we chose. Orlando has an FKGL of 23.78; although FKGL is not necessarily a good indicator of syntactic complexity, such a high FKGL does indicate that the sentences are

[2] `https://openai.com/pricing#language-models`

generally quite long. For comparison, *Time* magazine has an FKGL somewhere between 10 and 12.

We use the Princeton University Relation Extraction System (PURE) [60] on both the original and simplified versions of Orlando text. PURE does relation extraction in two stages: first it recognizes named entities, and then it extracts the relations between them. For each of these steps PURE uses a pre-trained BERT model [15]. We use PURE primarily because it has only 6 different types of relations and 7 different types of entities. This is convenient because the output of PURE needs to be reviewed manually since Orlando does not have the kind of annotations we need, but also because we would need to resort to manual alignment between the simplified and non-simplified text regardless of our choice of dataset. A small number of relations and entities makes the output of PURE easier to label.

We pre-process Orlando into the appropriate format for PURE and then use the scripts provided in the PURE GitHub repository[3] to run the system on the simplified and non-simplified Orlando data. Afterwards, a student manually labelled 100 relations (as correct or incorrect) extracted by PURE from the same simplified and non-simplified Orlando documents. Another student reviewed the labels and found them to be agreeable. In the non-simplified dataset 68 relations were correct, and in the simplified dataset 72 correct relations were found. Using Fisher's exact test, we find no statistically significant difference between the number of correct relations in both samples at the 0.01 significance level ($p=0.64$). We conclude that our AMR pipeline neither improves nor worsens the performance of PURE. It is unfortunate that our method did not improve performance on the relation extraction task, but it also seems that we do not lose any information which would prevent relation extraction, which is desirable.

5.2.2 Entity Linking

Our second extrinsic evaluation task is entity linking. The objective of entity linking is to disambiguate entities from unstructured text to known entities

[3] `https://github.com/princeton-nlp/PURE`

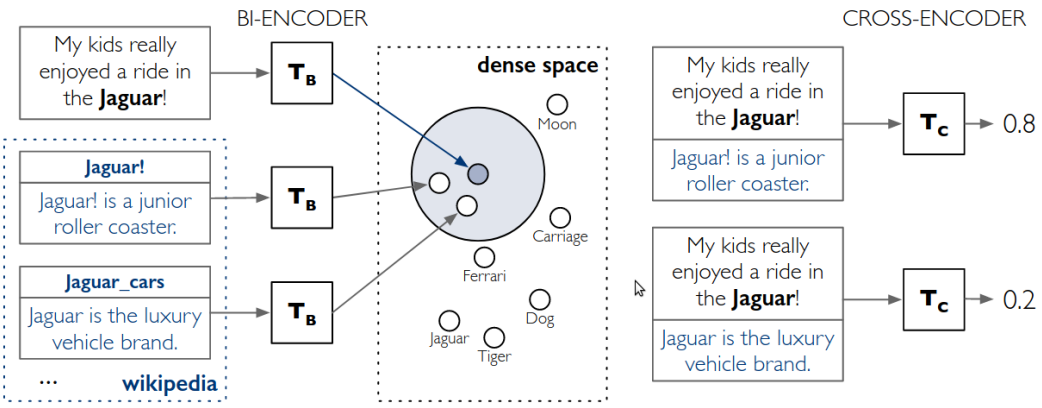

Figure 5.1: Description of BLINK entity linking procedure from Wu *et al.* [56]

in a knowledge base. For this evaluation we use the BLINK zero-shot entity linking solution which uses Wikipedia as the target knowledge base [56].

BLINK is yet another two-stage method which is based on BERT models. Figure 5.1, taken from the BLINK paper, depicts the overall approach. First, a biencoder based on BERT encodes all of the entities in the knowledge base into a dense space. Any new input is encoded with the same biencoder and the k nearest neighbors to the new entity are retrieved as candidate entities. Afterwards a cross-encoder (also based on BERT) ranks each of the candidate entities and selects the best match. This two-stage approach makes BLINK very fast; it would take a long time to compare a new entity to every entity in the knowledge base with a cross-encoder, but the biencoder with nearest neighbor search can narrow down the selection to just k candidates.

We use the same datasets as BLINK in our evaluation so that we can easily compare results. Additionally, the repository[4] contains scripts to download and pre-process all of the datasets except for one (the proprietary TAC-KBP 2010), which we exclude from the evaluation. Table 5.4 contains the list of datasets we use and the number of documents in each dataset. For each dataset we process every document with the AMR simplification pipeline and convert them to the appropriate input format for BLINK. We use the evaluation scripts in the BLINK repository to run the entity linker and generate results for the simplified text.

[4] `https://github.com/facebookresearch/BLINK/`

Dataset	Support
AIDA-YAGO2a	4766
AIDA-YAGO2b	4446
ACE 2004	244
aquaint	680
msnbc	617
WNED-WIKI	6383

Table 5.4: Size of the datasets used in BLINK evaluation.

	biencoder acc.		recall		cross-encoder normalized acc.		overall unnormalized acc.	
Dataset	**base**	**AMR**	**base**	**AMR**	**base**	**AMR**	**base**	**AMR**
AIDA-YAGO2a	0.81	0.82	0.98	0.99	0.87	0.82	0.82	0.81
AIDA-YAGO2b	0.8	0.81	0.97	0.99	0.87	0.81	0.8	0.8
ACE 2004	0.84	0.82	0.98	0.96	0.89	0.88	0.87	0.85
aquaint	0.87	0.87	0.99	0.98	0.89	0.87	0.86	0.85
msnbc	0.84	0.84	0.97	0.98	0.9	0.81	0.85	0.89
WNED-WIKI	0.8	0.78	0.98	0.97	0.86	0.81	0.81	0.79

Table 5.5: BLINK entity linking results. Each metric is reported from running BLINK on the base datasets and the text processed with the AMR pipeline.

The results with all the metrics produced by the evaluation script are in Table 5.5. For each metric, BLINK's performance on the original text is in the "base" column, and the performance on the simplified text is in the "AMR" column. The most important metric for this experiment is the overall unnormalized accuracy, found in the last column of Table 5.5. The unnormalized accuracy is just counting the number of entities correctly linked divided by the total number of entities. This is not the same as the number of documents correctly linked because every document can have multiple entities.

Comparing BLINK's performance on the simplified text to the original text, we find that there is no statistically significant difference at the 0.01 significance level in the overall accuracy achieved by BLINK between the two types of text on all datasets except for one. On the WNED-WIKI dataset BLINK has a slightly lower accuracy which is statistically significant ($p=0.005$). We conclude that there may be some degradation in the performance of BLINK from using AMR-simplified text, but it is slight. This indicates that our simplified text does not lose very much information that is useful for entity linking.

After seeing that BLINK's performance did not significantly degrade from

	biencoder acc.		recall		cross-encoder normalized acc.		overall unnormalized acc.	
Dataset	**base**	**AMR**	**base**	**AMR**	**base**	**AMR**	**base**	**AMR**
AIDA-YAGO2a	0.81	0.56	0.98	0.68	0.87	0.83	0.82	0.56
AIDA-YAGO2b	0.8	0.56	0.97	0.67	0.87	0.84	0.8	0.56
ACE 2004	0.84	0.67	0.98	0.78	0.89	0.91	0.87	0.71
aquaint	0.87	0.62	0.99	0.72	0.89	0.87	0.86	0.63
msnbc	0.84	0.63	0.97	0.73	0.9	0.88	0.85	0.64
WNED-WIKI	0.8	0.53	0.98	0.67	0.86	0.83	0.81	0.56

Table 5.6: BLINK entity linking results with target knowledge base embeddings based on text simplified with our AMR pipeline. Each metric is reported from running BLINK on the base datasets and the text processed with the AMR pipeline.

linking entities from our simplified text, we wondered whether we could improve BLINK's performance by also simplifying the entity descriptions that are used by the biencoder to create embeddings and by the cross-encoder to compare pairs of entities. The idea is that if the input text is more similar to the entity descriptions in the knowledge base, performance could improve. BLINK's target knowledge base consists of 5.9 million entities from a full Wikipedia dump. We ran the AMR simplification pipeline on all 5.9 million entity descriptions, which took approximately two weeks on 8 NVIDIA A40 GPUs. Afterwards, we re-ran BLINK's biencoder on the simplified text to create new embeddings for the entities, which took another few days. The result is that the biencoder is now embedding new simplified inputs into a dense space generated from simplified text, rather than trying to embed simplified text into a dense space created from standard Wikipedia descriptions. We run the BLINK evaluation again with the new embeddings; the results are in Table 5.6.

Unfortunately, we found that the overall accuracy is significantly worse when using the simplified embeddings compared to the standard embeddings. This is primarily because the biencoder does not generate an embedding for new entities which is close to the true entity in the knowledge base. The recall (second column Table 5.6) indicates how frequently the correct entity is in the top k=100 nearest entities to the embedding generated by the biencoder, and it is significantly lower for all of the datasets compared to the recall in Table

	cross-encoder accuracy	
Dataset	**regular embeddings**	**simplified embeddings**
AIDA-YAGO2a	0.82	0.83
AIDA-YAGO2b	0.81	0.84
ACE 2004	0.88	0.91
aquaint	0.87	0.87
msnbc	0.81	0.88
WNED-WIKI	0.81	0.83

Table 5.7: BLINK cross-encoder accuracy comparison between entity linking experiments with the original knowledge base and a knowledge base based on simplified text. These are the same cross-encoder numbers from Table 5.5 and Table 5.6.

5.5. Despite worse performance from the biencoder, the cross-encoder actually performs slightly better. Table 5.7 compares the cross-encoder accuracy when using the regular embeddings to our simplified embeddings. The cross-encoder accuracy is how often the cross-encoder chooses the correct enitity from the top k selected by nearest neighbor search. These results support the idea that the decreased performance is due to the biencoder, though we do not know exactly why. Perhaps the performance could be improved by finetuning the biencoder or using a different model altogether.

Overall, we believe the entity linking results indicate that our simplified text does not lose very much information compared to the un-simplified text. In the first experiment, where we only simplified the input text, there was no statistically significant decrease in BLINK's overall accuracy except for one dataset where the decrease was small. In the second experiment we simplified the entire knowledge base there was a large and statistically significant decrease in accuracy. However, the decrease can be explained by the biencoder's degraded performance. The fact that the cross-encoder performed slightly better suggests that the simplified text is still useful for linking, but the embeddings from the biencoder seem to be low quality.

Chapter 6

Limitations & Threats to Validity

In this chapter we examine factors that may contribute to our method producing bad results. We also discuss threats to the validity of our conclusions, claims resulting from our methods, and procedures.

6.1 Limitations of AMR

The foundation of this work is AMR, and like Banarescu *et al.* [5] mention, AMR is designed for English and is not interlingua. Nonetheless, there is work on adapting AMR to different languages [29], [55], but much of the AMR work is still biased towards English. Fan and Gardent [16] shows promising results in using AMR for multilingual AMR-to-text generation, albeit with diminished performance in languages other than English. Our impression is that our method may also work for other languages, but perhaps not as well as for English. If there were competitive AMR parsers and AMR-to-text models for other languages, it is possible our method may work well, and perhaps without any changes to the sub-graph selection algorithm.

The AMR specification comes with its own limitations which may hinder the effectiveness AMR-to-text models. Banarescu *et al.* [5] mention several:

1. AMR does not represent inflectional morphology for tense and number. Inflectional morphemes are short segments of language added to words to give them some grammatical property, like tense, possession, or com-

parison. The absence of inflectional morphology for tense might mean that text generated from AMR could have awkward or incorrect use of tense.

2. AMR omits articles, though the AMR-to-text model seems to handle this well.

3. AMR has no universal quantifier.

4. There may be some ambiguity introduced by using PropBank framesets to represent all verbs.

Banarescu *et al.* [5] suggest that the lack of inflectional morphology for tense and number and the lack of articles could be addressed by layering in a lightweight syntactic-style representation, but to our knowledge there are aren't any implementations of such a thing, and if there are then AMR parsers and AMR-to-text models which support the standard AMR specification would need to be retrained to support these additions.

6.2 Other Limitations

Consider again the following sentence:

> In 1935 they moved back to Chaldon, to live in a primitive stone cottage known as 24 West Chaldon.

Which our method simplifies into the following three sentences:

- They lived in primitive stone cottages.
- They moved back to Chaldon to live in 1935.
- The primitive stone cottages were known as 24 West Chaldon.

There is some information in the original sentence which is lost in the simplification. Specifically, the cardinality of the cottage is wrong (it should be singular, not plural), and there is some ambiguity about when the subject

moved and lived in Chaldon. In the original sentence, the subject lived in Chaldon *after* 1935, but the simplified sentences may give the impression that the subject only lived there *in* 1935. These issues may have to do with the underlying AMR representation.

Another practical limitation of our method is that our notion of simplification is purely syntactical. We simplify complicated sentences into smaller sentences based on the verbs. However, it may sometimes be advantageous (for fluency) to not simplify sentences which are already relatively simple, and where splitting would break causal or temporal dependencies in the text. Alternatively, the sentences could be simplified, but some information could be repeated in the sub-sentences (maybe even as a simple dependent clause) to preserve dependencies. We speculate that this alternative approach may produce better text from the perspective of humans, and also help NLP systems that look for local dependencies in text, such as pre-LLM question answering systems.

6.3 Threats to Validity

We have discussed how there is no consensus about which metrics to use for syntactic text simplification, and even commonly used metrics have been criticized. This may raise questions about the conclusions we draw from the intrinsic evaluation, but we believe the metrics we use are sufficient based on the fact that they are well-represented in the relevant literature. One thing that is absent from our experiments is a human evaluation of our simplifications, which many text simplification papers have. With a human evaluation, we might be able to make stronger conclusions about our method.

At the same time, human evaluations can introduce bias and human error. Our extrinsic evaluation on relation extraction in Section 5.2.1 presents results that depend on human-annotated data. We believe the results to be accurate, but there is the possibility that other human annotators might disagree with our annotations. Additionally, the relation extraction experiment might benefit from using a relation extraction system with more relations and entity

types; the relations extracted by PURE, while seemingly correct, are overly generic and not always very informative. A system with a more expressive set of relations has greater potential to uncover issues with our method.

In Section 5.2.2 we evaluate the performance of BLINK with the original knowledge base as well as the same knowledge base but with its text simplified using our method. The biencoder performs significantly worse on the simplified text, while the crossencoder performs better. We have a hard time explaining these results, and a satisfactory explanation might change our conclusions.

One of the baselines we compare to in Section 5.1 is ChatGPT-3.5. We manually experimented with some prompts that seemed to work, although admittedly we do not have a thorough quantitative evaluation of different prompts and how they change the quality and nature of the simplifications produced by ChatGPT-3.5. We believe the prompts we chose are fine, but it is possible that there are better prompts which could produce even higher quality simplifications. If this is the case, our method would be less competitive.

While we evaluate our method on downstream NLP tasks, we did not evaluate the output of ChatGPT-3.5 on these tasks. It is possible that BERT-based methods like the ones we test are not sensitive to the syntactic complexity of text, and the output of our method may be comparable to ChatGPT-3.5 for the purposes of downstream NLP tasks.

Chapter 7

Conclusion & Future Work

7.1 Future Work

Currently, our method only does syntactic simplification, but it could be extended to function as a full text text simplification pipeline with the addition of a lexical simplification component. Lexical simplification could be added as another step during the pipeline; other text simplification methods, both rule-based and neural, have had success with this approach [46], [59]. Another way to add lexical simplification might be to follow other neural methods which treat text simplification as a Seq2Seq task and use a single model end to end [30], [38]. In this case, AMRBART could be finetuned to learn a correspondence between AMR graphs and simplified text directly, though this approach might require a lot of training data.

The sub-graph selection algorithm we describe in Section 4.4 could be thought of as a type of rule which we use to simplify sentences. It is possible that our sentence splitting rule is not general enough, or that there are better ways to split sentences. One way to improve the sub-graph selection could be to train a model to select the splits instead of using our rule. For this task, reinforcement learning might be a good approach. Zhang and Lapata [59] use reinforcement learning to help with syntactic simplification, however the method they describe uses a model that learns token-level operations rather than operations over a hierarchical structure. To augment our method one could formulate the sub-graph selection problem as a reinforcement learning problem, and learn actions like when to split the AMR graph or add nodes to

a split.

Our syntactic simplification method currently depends on AMRBART for both AMR parsing and AMR-to-text generation. As the state of the art improves, either the AMR parser we use, or the AMR-to-text model, or both, could be replaced to potentially improve the performance of our pipeline. Lee *et al.* [27] introduce an AMR parser which outperforms AMRBART on some datasets, but also achieves state of the art performance for cross-lingual AMR parsing. Although strong AMR-to-text methods currently focus on English, eventually it could be worthwhile to experiment with using our method for other languages with the release of better models.

7.2 Conclusion

We present an AMR-based method for simplifying text and demonstrate its efficacy with an intrinsic and extrinsic evaluation. In the intrinsic evaluation we compare our method to DisSim, a best in class rule-based method, and ChatGPT-3.5, a strong LLM. We outperform DisSim and achieve performance comparable to ChatGPT-3.5 across various metrics. Unlike ChatGPT-3.5, our method has relatively low computational requirements. Our method can be hosted locally with modest hardware requirements, which has the advantage of reducing dependency on a third party service and preempting any privacy concerns associated with sending data to a model hosted off-site.

We perform an extrinsic evaluation on two downstream NLP tasks: entity linking and relation extraction. Using the text produced by our simplification pipeline for these downstream does not significantly degrade the performance of the methods we use for the evaluations. The extrinsic evaluation shows that our method does not lose very much information from the original text, and that the output is still suitable for downstream NLP tasks.

Overall, our intention is not to claim state of the art performance. There are currently stronger LLMs than ChatGPT-3.5 (like ChatGPT-4), which may have better performance on syntactic text simplification, though we could not find any literature explicitly investigating this. Additionally, the advantages

of our method could be short-lived; open source LLMs are catching up rapidly, and soon it may be possible to locally host models which can achieve comparable performance to ChatGPT variants across various NLP tasks, perhaps with relatively lower computational requirements, but likely still higher computational requirements than our method.

References

[1] O. Abend and A. Rappoport, "Universal Conceptual Cognitive Annotation (UCCA)," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, H. Schuetze, P. Fung, and M. Poesio, Eds., Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 228–238. [Online]. Available: https://aclanthology.org/P13-1023.

[2] S. Al-Thanyyan and A. M. Azmi, "Automated text simplification: A survey," *ACM Comput. Surv.*, vol. 54, no. 2, 43:1–43:36, 2022. DOI: 10.1145/3442695. [Online]. Available: https://doi.org/10.1145/3442695.

[3] F. Alva-Manchego, L. Martin, C. Scarton, and L. Specia, "EASSE: Easier automatic sentence simplification evaluation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 49–54. DOI: 10.18653/v1/D19-3009. [Online]. Available: https://aclanthology.org/D19-3009.

[4] X. Bai, Y. Chen, and Y. Zhang, "Graph pre-training for AMR parsing and generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6001–6015. [Online]. Available: https://aclanthology.org/2022.acl-long.415.

[5] L. Banarescu, C. Bonial, S. Cai, *et al.*, "Abstract Meaning Representation for sembanking," in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 178–186. [Online]. Available: https://aclanthology.org/W13-2322.

[6] J. A. Bateman, "Towards meaning-based machine translation: Using abstractions from text generation for preserving meaning," *Machine Translation*, vol. 7, no. 1/2, pp. 5–40, 1992, ISSN: 09226567, 15730573. [Online]. Available: http://www.jstor.org/stable/40011016 (visited on 10/29/2023).

[7] J. A. Botha, M. Faruqui, J. Alex, J. Baldridge, and D. Das, "Learning to split and rephrase from wikipedia edit history," *CoRR*, vol. abs/1808.09468, 2018. arXiv: 1808.09468. [Online]. Available: http://arxiv.org/abs/1808.09468.

[8] S. Brown, P. Clements, and I. Grundy, *Orlando: Women's writing in the british isles from the beginnings to the present*, 2022. [Online]. Available: https://orlando.cambridge.org.

[9] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. arXiv: 2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[10] R. Chandrasekar, C. Doran, and B. Srinivas, "Motivations and methods for text simplification," in *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*, 1996. [Online]. Available: https://aclanthology.org/C96-2183.

[11] Z. Cheng, Z. Li, and H. Zhao, "BiBL: AMR parsing and generation with bidirectional Bayesian learning," in *Proceedings of the 29th International Conference on Computational Linguistics*, N. Calzolari, C.-R. Huang, H. Kim, *et al.*, Eds., Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 5461–5475. [Online]. Available: https://aclanthology.org/2022.coling-1.485.

[12] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. arXiv: 1406.1078. [Online]. Available: http://arxiv.org/abs/1406.1078.

[13] A. Cossu, A. Carta, V. Lomonaco, and D. Bacciu, "Continual learning for recurrent neural networks: An empirical evaluation," *Neural Networks*, vol. 143, pp. 607–627, Nov. 2021, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.07.021. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2021.07.021.

[14] O. M. Cumbicus-Pineda, I. Gonzalez-Dios, and A. Soroa, "A syntax-aware edit-based system for text simplification," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, Held Online: INCOMA Ltd., Sep. 2021, pp. 324–334. [Online]. Available: https://aclanthology.org/2021.ranlp-1.38.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[16] A. Fan and C. Gardent, "Multilingual AMR-to-text generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 2889–2901. DOI: 10.18653/v1/2020.emnlp-main.231. [Online]. Available: https://aclanthology.org/2020.emnlp-main.231.

[17] Y. Feng, J. Qiang, Y. Li, Y. Yuan, and Y. Zhu, "Sentence simplification via large language models," *ArXiv*, vol. abs/2302.11957, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257102624.

[18] P. Gage, "A new algorithm for data compression," *The C Users Journal archive*, vol. 12, pp. 23–38, 1994. [Online]. Available: https://api.semanticscholar.org/CorpusID:59804030.

[19] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, "The WebNLG challenge: Generating text from RDF data," in *Proceedings of the 10th International Conference on Natural Language Generation*, J. M. Alonso, A. Bugarín, and E. Reiter, Eds., Santiago de Compostela, Spain: Association for Computational Linguistics, Sep. 2017, pp. 124–133. DOI: 10.18653/v1/W17-3518. [Online]. Available: https://aclanthology.org/W17-3518.

[20] G. Glavaš and S. Štajner, "Event-centered simplification of news stories," in *Proceedings of the Student Research Workshop associated with RANLP 2013*, I. Temnikova, I. Nikolova, and N. Konstantinova, Eds., Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, Sep. 2013, pp. 71–78. [Online]. Available: https://aclanthology.org/R13-2011.

[21] R. Hijazi, B. Espinasse, and N. Gala, "GRASS: A SYNTACTIC TEXT SIMPLIFICATION SYSTEM BASED ON SEMANTIC REPRESENTATIONS," in *11th International Conference on Natural Language Processing*, Copenhagen, Denmark, Sep. 2022. [Online]. Available: https://hal.science/hal-03865142.

[22] P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," 1975. [Online]. Available: https://api.semanticscholar.org/CorpusID:61131325.

[23] H. T. Lam, G. Picco, Y. Hou, *et al.*, "Ensembling graph predictions for AMR parsing," *CoRR*, vol. abs/2110.09131, 2021. arXiv: 2110.09131. [Online]. Available: https://arxiv.org/abs/2110.09131.

[24] I. Langkilde and K. Knight, "Generation that exploits corpus-based statistical knowledge," in *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998. [Online]. Available: https://aclanthology.org/C98-1112.

[25] J. Lee and J. B. K. P. Don, "Splitting complex English sentences," in *Proceedings of the 15th International Conference on Parsing Technologies*, Y. Miyao and K. Sagae, Eds., Pisa, Italy: Association for Computational Linguistics, Sep. 2017, pp. 50–55. [Online]. Available: https://aclanthology.org/W17-6307.

[26] Y. Lee, R. F. Astudillo, T. L. Hoang, T. Naseem, R. Florian, and S. Roukos, "Maximum bayes smatch ensemble distillation for AMR parsing," *CoRR*, vol. abs/2112.07790, 2021. arXiv: 2112.07790. [Online]. Available: https://arxiv.org/abs/2112.07790.

[27] Y. Lee, R. F. Astudillo, T. L. Hoang, T. Naseem, R. Florian, and S. Roukos, "Maximum bayes smatch ensemble distillation for AMR parsing," *CoRR*, vol. abs/2112.07790, 2021. arXiv: 2112.07790. [Online]. Available: https://arxiv.org/abs/2112.07790.

[28] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *CoRR*, vol. abs/1910.13461, 2019. arXiv: 1910.13461. [Online]. Available: http://arxiv.org/abs/1910.13461.

[29] B. Li, Y. Wen, L. Song, W. Qu, and N. Xue, "Building a Chinese AMR bank with concept and relation alignments," *Linguistic Issues in Language Technology*, vol. 18, Jul. 2019. [Online]. Available: https://aclanthology.org/2019.lilt-18.2.

[30] L. Martin, É. de la Clergerie, B. Sagot, and A. Bordes, "Controllable sentence simplification," English, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, May 2020, pp. 4689–4698, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.577.

[31] S. Narayan and C. Gardent, "Hybrid simplification using deep semantics and machine translation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Toutanova and H. Wu, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 435–445. DOI: 10.3115/v1/P14-1041. [Online]. Available: https://aclanthology.org/P14-1041.

[32] S. Narayan and C. Gardent, "Unsupervised sentence simplification using deep semantics," in *Proceedings of the 9th International Natural Language Generation conference*, A. Isard, V. Rieser, and D. Gkatzia, Eds., Edinburgh, UK: Association for Computational Linguistics, Sep. 2016, pp. 111–120. DOI: 10.18653/v1/W16-6620. [Online]. Available: https://aclanthology.org/W16-6620.

[33] S. Narayan, C. Gardent, S. B. Cohen, and A. Shimorina, "Split and rephrase," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 606–616. DOI: 10.18653/v1/D17-1064. [Online]. Available: https://aclanthology.org/D17-1064.

[34] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh, "DisSim: A discourse-aware syntactic text simplification framework for English and German," in *Proceedings of the 12th International Conference on Natural Language Generation*, Tokyo, Japan: Association for Computational Linguistics, Oct. 2019, pp. 504–507. DOI: 10.18653/v1/W19-8662. [Online]. Available: https://aclanthology.org/W19-8662.

[35] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh, "Transforming complex sentences into a semantic hierarchy," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3415–3427. DOI: 10.18653/v1/P19-1333. [Online]. Available: https://aclanthology.org/P19-1333.

[36] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh, "Transforming complex sentences into a semantic hierarchy," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3415–3427. DOI: 10.18653/v1/P19-1333. [Online]. Available: https://aclanthology.org/P19-1333.

[37] C. Niklaus, A. Freitas, and S. Handschuh, "MinWikiSplit: A sentence splitting corpus with minimal propositions," in *Proceedings of the 12th International Conference on Natural Language Generation*, Tokyo, Japan: Association for Computational Linguistics, Oct. 2019, pp. 118–123. DOI: 10.18653/v1/W19-8615. [Online]. Available: https://aclanthology.org/W19-8615.

[38] S. Nisioi, S. Štajner, S. P. Ponzetto, and L. P. Dinu, "Exploring neural text simplification models," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 85–91. DOI: 10.18653/v1/P17-2014. [Online]. Available: https://aclanthology.org/P17-2014.

[39] M. Palmer, D. Gildea, and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Comput. Linguist.*, vol. 31, no. 1, pp. 71–106, Mar. 2005, ISSN: 0891-2017. DOI: 10.1162/0891201053630264. [Online]. Available: https://doi.org/10.1162/0891201053630264.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://aclanthology.org/P02-1040.

[41] R. Pascanu, T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks*, 2013. arXiv: 1211.5063 [cs.LG].

[42] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," *Transactions of the Association for Computational Linguistics*, vol. 8, M. Johnson, B. Roark, and A. Nenkova, Eds., pp. 842–866, 2020. DOI: 10.1162/tacl_a_00349. [Online]. Available: https://aclanthology.org/2020.tacl-1.54.

[43] N. Sadvilkar and M. Neumann, "PySBD: Pragmatic sentence boundary disambiguation," in *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 110–114. [Online]. Available: https://www.aclweb.org/anthology/2020.nlposs-1.15.

[44] T. Sellam, D. Das, and A. P. Parikh, "Bleurt: Learning robust metrics for text generation," in *Proceedings of ACL*, 2020.

[45] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *CoRR*, vol. abs/1508.07909, 2015. arXiv: 1508.07909. [Online]. Available: http://arxiv.org/abs/1508.07909.

[46] A. Siddharthan and A. Mandya, "Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules," in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, S. Wintner, S. Goldwater, and S. Riezler, Eds., Gothenburg, Sweden: Association for Computational Linguistics, Apr. 2014, pp. 722–731. DOI: 10.3115/v1/E14-1076. [Online]. Available: https://aclanthology.org/E14-1076.

[47] E. Sulem, O. Abend, and A. Rappoport, "BLEU is not suitable for the evaluation of text simplification," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Association for Computational Linguistics, 2018, pp. 738–744. DOI: 10.18653/v1/d18-1081. [Online]. Available: https://doi.org/10.18653/v1/d18-1081.

[48] E. Sulem, O. Abend, and A. Rappoport, "Semantic structural evaluation for text simplification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Or-

leans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 685–696. DOI: 10.18653/v1/N18-1063. [Online]. Available: https://aclanthology.org/N18-1063.

[49] E. Sulem, O. Abend, and A. Rappoport, "Simple and effective text simplification using semantic and neural methods," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 162–173. DOI: 10.18653/v1/P18-1016. [Online]. Available: https://aclanthology.org/P18-1016.

[50] S. Surya, A. Mishra, A. Laha, P. Jain, and K. Sankaranarayanan, "Unsupervised neural text simplification," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2058–2068. DOI: 10.18653/v1/P19-1198. [Online]. Available: https://aclanthology.org/P19-1198.

[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. arXiv: 1409.3215. [Online]. Available: http://arxiv.org/abs/1409.3215.

[52] T. Tanprasert and D. Kauchak, "Flesch-kincaid is not a text simplification evaluation metric," in *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, A. Bosselut, E. Durmus, V. P. Gangal, *et al.*, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 1–14. DOI: 10.18653/v1/2021.gem-1.1. [Online]. Available: https://aclanthology.org/2021.gem-1.1.

[53] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[54] V. Vo, W. Wang, and W. L. Buntine, "Unsupervised sentence simplification via dependency parsing," *CoRR*, vol. abs/2206.12261, 2022. DOI: 10.48550/arXiv.2206.12261. arXiv: 2206.12261. [Online]. Available: https://doi.org/10.48550/arXiv.2206.12261.

[55] S. Wein, L. Donatelli, E. Ricker, *et al.*, "Spanish Abstract Meaning Representation: Annotation of a general corpus," in *Northern European Journal of Language Technology, Volume 8*, Copenhagen, Denmark: Northern European Association of Language Technology, 2022. DOI: https://doi.org/10.3384/nejlt.2000-1533.2022.4462. [Online]. Available: https://aclanthology.org/2022.nejlt-1.6.

[56] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Zero-shot entity linking with dense entity retrieval," in *EMNLP*, 2020.

[57] W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch, "Optimizing statistical machine translation for text simplification," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 401–415, 2016. DOI: 10.1162/tacl_a_00107. [Online]. Available: https://aclanthology.org/Q16-1029.

[58] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," *CoRR*, vol. abs/1904.09675, 2019. arXiv: 1904.09675. [Online]. Available: http://arxiv.org/abs/1904.09675.

[59] X. Zhang and M. Lapata, "Sentence simplification with deep reinforcement learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa, and S. Riedel, Eds., Association for Computational Linguistics, 2017, pp. 584–594. DOI: 10.18653/v1/d17-1062. [Online]. Available: https://doi.org/10.18653/v1/d17-1062.

[60] Z. Zhong and D. Chen, "A frustratingly easy approach for entity and relation extraction," in *North American Association for Computational Linguistics (NAACL)*, 2021.