TOWARDS A PRACTICALLY USEFUL TEXT SIMPLIFICATION SYSTEM

Reno Kriz

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2021

Dedicated to my parents, Thomas and Kathy Kriz.

ACKNOWLEDGEMENT

I first would like to thank my advisors Chris Callison-Burch and Marianna Apidianaki. When I came to Penn, I really did not have much experience in research, and cannot thank Chris enough for gently encouraging me in my and including me in different projects until I found a field that really spoke to me. As I gained experience, he always gave me the freedom to explore research I was interested in. Beyond research, Chris fostered a working environment that always made me feel supported, which helped me to slowly become more comfortable with just being myself. When Marianna came to Penn as a visiting researcher in our lab, she very quickly became a second mentor to me. She was always available to brainstorm ideas and help get me over the finish line for my first few research projects. I am so thankful that Marianna came to Penn at just the right time for me, and I am thrilled that we were able to continue collaborating, and that we could eventually make her an official co-advisor.

I am also grateful for the additional mentorship given to me by Eleni Miltsakaki and João Sedoc. Eleni brought me on as an intern at Choosito as a relatively inexperienced first-year PhD student in the summer of 2017, and was the one to introduce me to the field of text simplification. That internship both led to my first paper in a major conference, gave me my thesis topic, and led to collaborations on multiple other projects, so I can confidently say it was the most successful internship I have ever had. I was excited that she was able to join my committee, and that I could be a part of her transition back into the Penn research community. João, as a senior PhD student, could have easily focused on his own projects in his last couple years, but instead he chose to help me learn and explore neural text generation models. He was always available to provide answers when I was stuck with anything, whether it was a technical problem or a deeper research question.

I am thankful for the rest of my thesis committee - Mitch Marcus, Lyle Ungar, Dan Roth, and Eleni Miltsakaki - for helping me focus my thesis on what could really be the most helpful for the field. Mitch is incredibly knowledgeable both on how to manage the thesis process and just NLP field generally, and was always willing to take the time to learn about new ideas. Dan's neural NLP class sparked the ideas behind Chapter 4 of this thesis, and his lab has been an amazing addition and resource for the Penn NLP community. Lyle enthusiasm and insightful questions helped me learn about our field and better frame my research within it.

I am indebted to my other collaborators. On the papers discussed in this thesis I received so much help from Daphne Ippolito, Rebecca Iglesias-Flores, Ajay Patel, Carolina Zheng, Gaurav Kumar, Maria Kustikova, and Megha Mishra. I would also like to thank all of the students, postdocs, and researchers that I was lucky enough to be around every day: Ellie, Derry, Andy, Jie, Jordan, Devanshu, Nitish, Shyam, Stephen, Hangfeng, Dan, Daniel, Danni, Aditya, Lara, Mohammad, Harry, Veronica, Bryan, Alyssa, and Samar.

Beyond people at Penn, I want to thank my friends and family for supporting me through these five years. To my parents, Thomas and Kathy, thank you for giving me this opportunity, and for going through my research papers even when you did not fully understand them. To my sister Shannon, my cousin Cassie, and my friends Colin, Malik, Osiris, Adrian, Pedro, Olivia, Alice, Andre, Rob, Ricci, Kevin, Max, Matt, Eliza, and Loki, thank you all for helping me relax and appreciate my time in grad school. To my friends To my roommate Matteo, seeing you working to finish your PhD and find a job right next to me, despite a pandemic changing our world for the last year, really inspired me to keep going and reach my own finish line too.

ABSTRACT

TOWARDS A PRACTICALLY USEFUL TEXT SIMPLIFICATION SYSTEM

Reno Kriz

Chris Callison-Burch and Marianna Apidianaki

While there is a vast amount of text written about nearly any topic, this is often difficult for someone unfamiliar with a specific field to understand. Automated text simplification aims to reduce the complexity of a document, making it more comprehensible to a broader audience. Much of the research in this field has traditionally focused on simplification sub-tasks, such as lexical, syntactic, or sentence-level simplification. However, current systems struggle to consistently produce high-quality simplifications. Phrase-based models tend to make too many poor transformations; on the other hand, recent neural models, while producing grammatical output, often do not make all needed changes to the original text. In this thesis, I discuss novel approaches for improving lexical and sentence-level simplification systems. Regarding sentence simplification models, after noting that encouraging diversity at inference time leads to significant improvements, I take a closer look at the idea of diversity and perform an exhaustive comparison of diverse decoding techniques on other generation tasks. I also discuss the limitations in the framing of current simplification tasks, which prevent these models from yet being practically useful. Thus, I also propose a retrieval-based reformulation of the problem. Specifically, starting with a document, I identify concepts critical to understanding its content, and then retrieve documents relevant for each concept, re-ranking them based on the desired complexity level.

TABLE OF CONTENTS

LIST OF TABLES

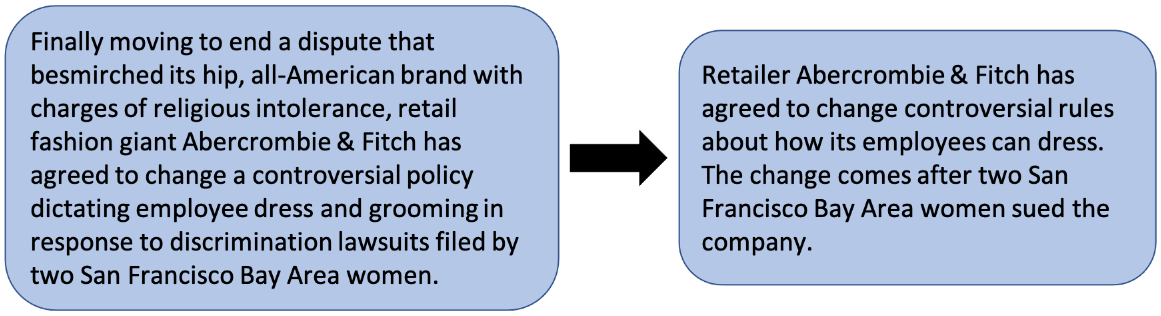
LIST OF ILLUSTRATIONS

CHAPTER 1 : Introduction

1.1. Overview

In the modern era, anyone can go online, search for any topic in which they are interested, and immediately find hundreds, if not thousands, of documents related to that topic. The current power of technology gives people the possibility to learn about almost anything, at least in theory. In reality however, it is often the case that documents are written in a relatively complex way, and/or require prior knowledge in a particular field before truly understanding the material. This is especially problematic for children, but also is an issue for adults with learning disabilities, foreign language learners, or other adults simply trying to learn about a completely new field. To solve this problem, we can consider the task of text simplification.

Automated text simplification is the process that involves transforming a complex text into one with the same meaning that can be more easily read and understood by a broader audience (Saggion, 2017). This process can include several sub-tasks such as lexical and phrasal simplification (Devlin and Tait, 1998; Shardlow, 2013b), reordering (Chandrasekar and Bangalore, 1997; Feblowitz and Kauchak, 2013), deletion, and sentence splitting (Zhu et al., 2010).



Finally moving to end a dispute that besmirched its hip, all-American brand with charges of religious intolerance, retail fashion giant Abercrombie & Fitch has agreed to change a controversial policy dictating employee dress and grooming in response to discrimination lawsuits filed by two San Francisco Bay Area women.

Retailer Abercrombie & Fitch has agreed to change controversial rules about how its employees can dress. The change comes after two San Francisco Bay Area women sued the company.

Figure 1: An example of an ideal output from a text simplification system. On the left, we show a long and complex text. On the right, we show a simplified version of this text, which preserves the meaning of the original text.

As a grounding example that shows these sub-tasks in practice, let's consider the complex text shown on the left of Figure 1 and its simplified version written by professional editors from Newsela[1], shown on the right. As examples of phrasal simplification, we see that *policy* was changed to *rules*, and *in response to* was changed to *after*. As a more syntactic modification, the final part of the complex text, *discrimination lawsuits filed by two San Francisco Bay area women*, was re-arranged to be presented in the active voice, *two San Francisco Bay area women sued the company.* The beginning of the complex text, *Finally moving to end a dispute that besmirched its hip, all-American brand with charges of religious intolerance*, has been removed, as this part does not contribute much to the meaning of the original text. Finally, we can see that the main part of the complex text has been split into two sentences; this required the addition of a clause, *The change comes*, in order to preserve grammaticality in the simplified version.

Many recent approaches attempt to formulate text simplification as a monolingual generation task and apply sequence-to-sequence (Seq2Seq) models (Nisioi et al., 2017; Zhang and Lapata, 2017; Zhao et al., 2018). These have been shown to give state-of-the-art results in machine translation (Sutskever et al., 2014), dialogue systems (Vinyals and Le, 2015), and many other natural language processing fields. While these methods have shown improvement over previous statistical and phrase-based approaches, their practical uses are still somewhat limited. There are several reasons for this. First, although recent simplification models generate grammatical output, they do not make all necessary changes to the original text (Zhang and Lapata, 2017). Second, there is a lack of quality simplification data available for training (Xu et al., 2015). Finally, current evaluation metrics have low or negative correlations with human judgments of simplification quality (Xu et al., 2016; Sulem et al., 2018). Beyond these issues, a higher-level problem is that these models view simplification as translating from a specific complex to a specific simple level. However, texts can be written at many complexity levels, and even the same text can be of varying

[1]Newsela is a commonly used simplification corpus introduced by Xu et al. (2015), discussed more in Section 2.3.2.

difficulty depending on a reader's background knowledge. Moreover, a text might contain domain-specific terminology which requires a more detailed explanation to be understood, rather than merely substituting it with a simpler term.

1.2. Thesis Statement

In this thesis, we claim that the textual complexity at any level is not a static value, but instead is influenced both by the surrounding text and especially the knowledge of a particular reader. Thus, in order to create a practically useful simplification system, it is critical to be able to adjust the outputs of our models depending on the situation.

In the chapters that follow, we demonstrate how this framing can inform and improve simplification systems at three levels: at the word level, where we show how to identify complex words and replace them with simpler substitutes while better taking the surrounding context into account; at the sentence level, where we show how to produce simpler outputs by generating a large number of diverse candidate sequences; and at the document level, where we consider how to identify concepts that are critical to understanding a document, and then how to retrieve related documents from a variety of complexity levels.

1.3. Outline of this Document

The rest of this document is organized as follows.

Chapter 2

We begin with an overview of related work in the field of text simplification, and include a general discussion of generation and language modeling. First, we present previous work specifically focusing on the lexical simplification task, which includes two sub-tasks: identifying complex words in a text (known as complex word identification or CWI), and replacing them with simpler substitutes. We also present several widely-used paraphrase datasets, which can be effectively leveraged for lexical simplification. Next, we briefly discuss corpora used for training sentence simplification systems. We then review several state-of-the-art approaches for training a holistic statistical simplification system, where the input is a complex English sentence and the output is a simple English sentence. The chapter continues with a discussion of Sequence-to-Sequence models, their applications to text simplification, and how to effectively generate outputs at inference time. Subsequently, we discuss in detail several commonly-used metrics for evaluating these systems; in this section, we also discuss the limitations of these metrics, and why human evaluation is still considered the most trustworthy method of evaluation simplification systems. We conclude this section with a discussion of large-scale pre-trained language models.

Chapter 3

In Chapter 3, we present our proposed approach to the task of lexical simplification. We propose to solve this task in two steps: we first identify words that are difficult to understand within a text, and then replace them with simpler alternatives that make sense in the same context.

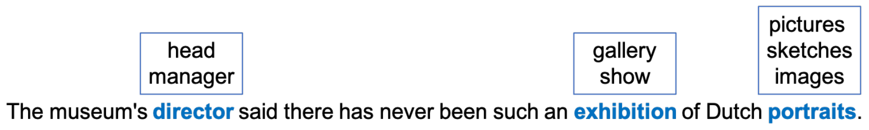

head
manager
gallery
show
pictures
sketches
images
The museum's director said there has never been such an exhibition of Dutch portraits.

Figure 2: An example sentence with complex words identified by our classifier, and substitutes proposed by our embedding-based lexical simplification model.

Specifically, in order to identify complex words, we train a Support Vector Machine (SVM) classifier which uses both word-based and context-based features. To train this classifier, we create a corpus of *in-context* complex and simple words annotated by Amazon Mechanical Turk workers. For the substitution step, we extract candidate substitutes from a large-scale database of simplifying paraphrase rules (Pavlick and Callison-Burch, 2016). We select the substitutes that best fit each context using a word embedding-based lexical substitution model. We show that our CWI classifier and lexical simplification model yield results that outperform several baseline approaches, and provide a detailed error analysis to show when our methods fall short. This chapter expands on our NAACL 2018 long paper on lexical simplification (Kriz et al., 2018).

Chapter 4

A simplification model that is useful in practice should not only make lexical or phrasal substitutions, but also perform other operations, such as reordering and deletion. In Chapter 4, we consider the task of sentence simplification, which aims to holistically simplify entire sentences. In this task, we start with a complex English sentence and attempt to generate a simple English sentence that is fluent (i.e. grammatical), adequate (i.e. meaning-preserving), and simpler than the original. In this part of our work, we leverage a Sequence-to-Sequence (Seq2Seq) model, which learns how to perform all simplification operations simultaneously.

Vanilla Sequence-to-Sequence (Seq2Seq) models (Sutskever et al., 2014) have been shown to work well on this task (Nisioi et al., 2017), although they often make very few changes to the original complex sentence (Zhang and Lapata, 2017). We address this problem by proposing extensions to the generic Seq2Seq framework at training, inference, and post-inference time. During training, we propose a custom loss function that rewards the model more for correctly generating simpler content words. We do so by multiplying the standard cross entropy loss function by the predicted complexities of each word in our vocabulary. At inference time, rather than using the standard greedy search to generate a single system output, we implement a diverse beam search approach which penalizes candidates that come from the same partial parent output, thus allowing us to explore more of the search space. Finally, at post-inference time we re-rank the multiple generated candidates using external methods to judge the fluency, adequacy, and simplicity of each output. An example output of our model compared with that generated from a vanilla Seq2Seq model is shown in Figure 3. While these extensions do result in simpler output sentences, there is a trade-off where we sometimes lose critical information from the original. We again provide an error analysis to discuss limitations of our approach in more detail. This section discusses work from our

NAACL 2019 long paper on sentence simplification (Kriz et al., 2019).

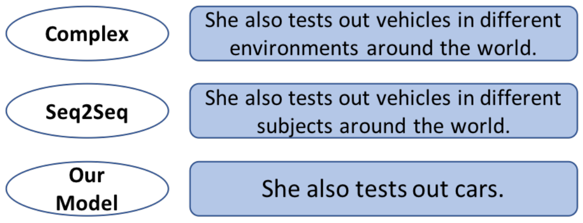

Complex
She also tests out vehicles in different environments around the world.
Seq2Seq
She also tests out vehicles in different subjects around the world.
Our Model
She also tests out cars.

Figure 3: Comparison of a simplification generated by a vanilla Seq2Seq model vs. our proposed model.

We observed that while our sentence simplification method yielded higher performance than previous methods, as measured using standard automatic simplification metrics, the results were more mixed after collecting manual human annotations. Thus, in the last part of this chapter, we discuss the creation of an automatic quality estimation metric that correlates better with human judgments, without requiring a reference sentence for comparison. Given a complex sentence and the corresponding simplified output, we fine-tune BERT (Devlin et al., 2019) to predict its fluency, adequacy, and relative complexity simultaneously. We show that training a single model rather than three separate models results in better performance, likely due to the fact that each individual dimension is somewhat correlated with the others. This section discusses work from our ArXiv short paper on simplification evaluation (Kriz et al., 2020).

Chapter 5

In Chapter 4, we implemented several extensions to Seq2Seq models for the task of text simplification; from these extensions, encouraging diversity at inference time led to significant improvements. This notion of diversity is not unique to text simplification, in fact many NLP generation tasks, including conversational dialogue systems (i.e. chatbots), image captioning, and story generation, can benefit from leveraging diverse candidate outputs for a single input. In Chapter 5, we take closer look at the idea of diversity, and perform an exhaustive comparison of current diverse decoding techniques. In this chapter, we define

Beam Search
A bus is stopped at a bus stop.
A bus is parked at a bus stop.
A bus stopped at a bus stop in a city.
A bus stopped at a bus stop at a bus stop.
A bus that is parked in front of a building.

Random Sampling
A bus parked at a bus stop at a bus stop.
There is a bus that is at the station.
A man standing by a bus in a city.
A bus pulling away from the train station.
A bus stopped at a stop on the sunny day.

Figure 4: An image with the top five captions produced by standard beam search and by random sampling. Note that the latter set is more diverse but of lower quality.

diversity as the ability of a generative method to create a set of possible outputs that are all valid for a given input, but vary as much as possible in terms of word choice, topic, and meaning.

Early neural machine translation work found that beam search is an effective heuristic to sample likely sequences from conditional language models (Sutskever et al., 2014). However, this was initially restricted to tasks where only the most likely output sequence was considered; if we want to use beam search to generate multiple candidates, it has been shown that these tend to only differ in punctuation and/or minor morphological variations (Li and Jurafsky, 2016). A variety of alternatives and extensions to beam search have thus been proposed for generating diverse candidate responses (Li et al., 2016a; Vijayakumar et al., 2016; Kulikov et al., 2018; Tam et al., 2019). Many of these approaches show marked improvement in diversity over standard beam search across a variety of generative tasks. However, there has been little comparison and evaluation of these strategies against each other on a single task. In this work we systematically compare existing diverse decoding methods on two tasks: chatbots and image captioning. We also propose the use of over-sampling followed by re-ranking at post-inference time.

In our experiments, we show that if the underlying model performs relatively well on a task, leveraging the increased diversity in a random sampling-based approach will likely lead to high-quality and diverse outputs. On the other hand, if the underlying model performs relatively poorly on the initial task, encouraging too much diversity will likely come at the cost of output quality. This work is based on work from our ACL 2019 long paper on diversity in conditional language models (Ippolito et al., 2019); with this paper, Daphne Ippolito and I were co-first authors, and share equal credit for this work.

Chapter 6

In Chapters 3 and 4 our proposed methodologies for lexical- and sentence-level simplification yield better results than previous approaches; however in our error analyses we identified several fluency and adequacy errors that our models still make quite often. At the sentence level, the errors are particularly pronounced the input sentences are long and complex. Fine-tuning pre-trained language generation models (Radford et al., 2019; Brown et al., 2020) could serve to better address some quality issues at the sentence level. However, these models would still fall short at the document level, as the text they generate still contains many long-term inconsistencies (Ippolito et al., 2020). Thus, it is currently difficult to scale generation further in order to simplify entire documents without significant loss in meaning.

To address these issues, Chapter 6 proposes to reformulate the text simplification task. Given a document D on some topic, we can 1) identify the concepts that are critical for understanding D, 2) retrieve a set of documents D' related to each concept from a large general corpus, where D' contains documents from a variety of complexity levels, and 3) re-rank D' to find documents written at levels that are most appropriate for the user. This is an extremely ambitious reformulation; thus the experiments in this Chapter are meant to serve as steps taken towards this final goal.

In the first section of Chapter 6, we discuss our methodology for identifying the critical concepts in a document D. There has been an extensive amount of work on keyphrase extraction, a field related to critical concept identification, so first we discuss the most prominent unsupervised methods that have been proposed to identify keyphrases in a single document. Furthermore, we present several simple Wikipedia-based baselines, which extract a set of domain-specific concepts by leveraging the internal Wikipedia category hierarchy. Finally, we describe recent BERT-based keyphrase extraction approaches, and propose a novel BERT-based approach that first identifies several high-quality seed concepts, before iteratively identifying additional concepts based on their similarity to the seed concepts. In order to evaluate these methods on real data, we create a test set consisting of computer science web articles. From there, we collect annotations for each article, and perform a secondary round of adjudication to ensure agreement. When tested on this evaluation set, BERT-based methods interestingly perform worse than previous unsupervised approaches that do not leverage large-scale pre-trained language models.

In the second section of Chapter 6, we focus on the tasks of retrieving related documents at different complexity levels ranked based on embedding similarity, and their re-ranking according to the desired complexity level. For this task, our methodology relies on a mechanism that combines embedding-based text similarity (Reimers and Gurevych, 2019) with a complexity re-ranking module. We conduct experiments in settings of varying difficulty, involving a diverse number of distractor documents. Initially, we leverage Newsela, a parallel corpus of 1,882 news articles rewritten at 5 complexity levels. Given a Newsela article written at complexity level 4 (L4), we attempt to retrieve the rewritten articles from L0, L1, L2, and L3, using the rest of the articles as distractors.

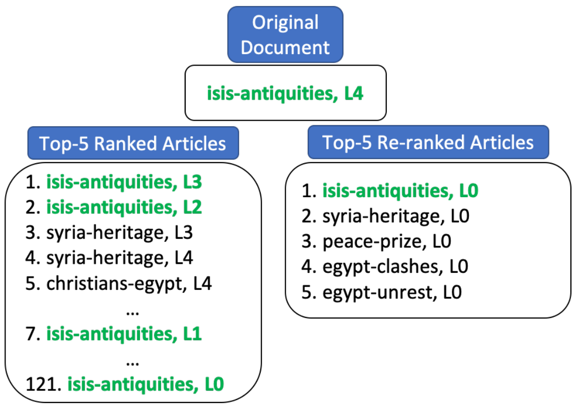


Figure 5: Given an input article at complexity Level 4 (L4), we show the ranking of aligned articles based on SBERT embedding similarity Reimers and Gurevych (2019), and the output of our re-ranking methodology which boosts articles at the lowest complexity level (L0).

We show that we can effectively retrieve L3 and L2 documents, as these are the versions with the least amount of changes and, thus, the most similar. However, current embeddings-based retrieval approaches seem to struggle with L1 and especially with L0 documents. To address this, we introduce a re-ranking mechanism which involves fine-tuning BERT to predict document-level complexity (Devlin et al., 2019). This mechanism allows us to effectively filter out documents at incorrect complexity levels, making our retrieval approach significantly more effective. Finally, in a large-scale experiment, we show that our methodology successfully retrieves related documents at the desired complexity level even in a challenging scenario with over one million candidate documents.

Chapter 7

Chapter 7 summarizes the key ideas and take aways from this thesis, along with its limitations and areas for future work.

CHAPTER 2 : Background and Related Work

2.1. Introduction

In this chapter, we will discuss related works of various subfields of English text simplification, as well as text generation more generally. This chapter is broken down into six main sections. First, Section 2.2 discusses lexical simplification, the most well-researched subtask of text simplification involving identifying complex words within a text, and replacing them with simpler words that preserve the meaning of the original sentence. This section also discusses various commonly-used resources used for lexical and phrasal substitution and simplification. This section is most relevant for the work involved in Chapter 3, though the idea of understanding textual complexity is something that is discussed throughout this thesis.

Next, Section 2.3 focuses on the main text simplification corpora leveraged by these sentence-level statistical systems, and later neural-based systems as well, to learn common simplification operations. In particular, we consider Parallel Wikipedia, which aligns sentences from Simple Wikipedia articles with their corresponding Wikipedia articles (Zhu et al., 2010); and also Newsela, a dataset of news articles re-written at five complexity levels by professional editors (Xu et al., 2015). These corpora, especially Newsela, are critical in many paraphrasing, generation, and retrieval experiments in Chapters 3, 4, and 6, respectively.

Section 2.4 focuses on phrase-based and statistical text simplification systems. These systems still attempt to address lexical and phrasal substitution, but also attempt to address other common subtasks such as deletion, reordering, and sentence splitting. These models often create statistically-driven components for each subtask, before combining them together into a pipeline to holistically simplify a sentence. This work is generally considered the state-of-the-art approach to text simplification prior to the rise of neural-based methods. Note that this is a trend common to machine translation, dialogue systems, and many other generation tasks.

After this, Section 2.5.1 focuses on the rise of neural-based approaches, both within text simplification and across natural language processing. We first provide a high-level description of the structure of a basic Sequence-to-Sequence (Seq2Seq) model (Sutskever et al., 2014), which typically involves an initial model that *encodes* an input into some latent representation, and a second model that takes this representation and uses it to *decode* an output one token at a time. Next, we discuss several applications of Seq2Seq approaches to sentence simplification, which we use as a basis of comparison for our own sentence simplification models in Chapter 4. Finally, we discuss several standard approaches for effectively decoding an output from a Seq2Seq model. These include greedy search, which involves simply taking the most likely token from the probability distribution at each time step; and beam search, which involves keeping track of the k most likely partial sequences throughout the decoding process. This section serves as a basis for the various diverse decoding approaches discussed some in Chapter 4, but especially in Chapter 5.

Section 2.6 considers various methods for evaluating sentence simplification systems. We first discuss the three main dimensions upon which systems are evaluated: fluency, adequacy, and relative complexity. We consider several metrics that attempt to approximate human judgments on these three dimensions when comparing to a single reference sentence or multiple reference sentences. After this, we also consider quality estimation metrics, which attempt to estimate the quality of a generated simple sentence *without a reference*. These metrics are used significantly in Chapter 4, both as ways to evaluate the quality of our own sentence simplification models, as well as the basis of comparison for the creation of our own metric.

Lastly, Section 2.7 briefly discusses large-scale pre-trained language models that have revolutionized nearly every aspect of natural language process in recent years. In particular, we focus on BERT, a contextual word embedding model trained on two very general language understanding tasks with a vast amount of data (Devlin et al., 2019). We discuss the importance of these language models on their own, but that the most noteworthy property of these models is that they can be fine-tuned on a small amount of task-specific data to significantly outperform models trained from scratch. We refer to BERT-based models throughout all Chapters: in Chapter 3 we perform a post-hoc comparison of our models to RoBERTa-based approaches; in Chapter 4 we fine-tuned BERT to estimate the quality of sentence simplification systems; in Chapter 5 we use BERT to cluster similar sentences together; and in Chapter 6 we leverage BERT-based models both for identifying critical concepts, for predicting document-level complexity, and ultimately for retrieving related documents at various complexity levels.

2.2. Lexical Simplification

2.2.1. Disambiguation-based and Direct Methods

Lexical simplification is a sub task of text simplification, which involves replacing complex words in a text with simpler substitutes that preserve their meaning. Lexical simplification can involve identifying complex words in context (Shardlow, 2013b), substitute generation (Biran et al., 2011), and finally choosing the appropriate substitute for each complex word.

To identify the words to be simplified, Shardlow (2013a) proposes to use a Support Vector Machine (SVM) that exploits several lexical features, such as token length, token frequency, number of characters, and number of syllables. Our best approach in Chapter 3 also integrates a SVM classifier for identifying complex words, but complements this set of features with context-related and embedding-based features that have not been exploited in previous work.

In lexical simplification, existing methods differ as to whether they include a word sense disambiguation (WSD) step for substitute selection, and in the ranking method used. Ranking is often performed based on word frequency in a large corpus, since it has been shown that frequent words increase a text's readability (Devlin and Tait, 1998; Kauchak, 2013). Models that include a semantic processing step for substitute selection aim to ensure that the selected substitutes express the correct meaning of words in specific contexts. WSD is often carried out by selecting the correct synset (i.e. set of synonyms describing a sense) for a target word in WordNet; the synonyms in the synset are then used as substitutes. Thomas and Anderson (2012) use WordNet's tree structure to reduce the size of the vocabulary in a document; we discuss WordNet and other paraphrase corpora in detail in the next section. Biran et al. (2011) perform disambiguation in an unsupervised manner. They learn simplification rules from comparable corpora and apply them to new sentences using vector-based context similarity measures to select words that are the most likely candidates for substitution in a given context. This process does not involve an explicit WSD step, and simplification is addressed as a context-aware lexical substitution task. The SemEval 2012 English Lexical Simplification task (Specia et al., 2012) also addresses simplification as lexical substitution (McCarthy and Navigli, 2007), allowing systems to use external sense inventories or to directly perform in-context substitution.

2.2.2. Lexical Semantic Resources

One of the earliest and best-known lexico-semantic resources is WordNet, a manually curated lexical network which encodes semantic relationships between words (Miller, 1995). WordNet contains 155,327 words grouped into 175,979 synsets, i.e. synonym sets, along with relationships between these synsets. Relationships between two synsets S_1 and S_2 include hypernymy, which occurs when the words in S_1 are subtypes of those in S_2; meronymy, which occurs when the words in S_1 denote parts of those in S_2; and antonymy, where the words in S_1 have opposite meaning from those in S_2. In addition, some words can appear in multiple distinct synsets, indicating that they have multiple meanings. For example, the word *bug* is related to nouns such as *insect* and *beetle*, but also separately to nouns such as *error* and *mistake* (Cocos and Callison-Burch, 2016).

Semantic resources have also been created using automatic methods. One such resource is the Paraphrase Database (PPDB), which contains more than 220 million English paraphrase pairs (Ganitkevitch et al., 2013; Pavlick et al., 2015). PPDB was created using a bilingual pivoting technique, which assumes that two English phrases that translate to the same foreign phrase have the same meaning (Bannard and Callison-Burch, 2005). This pivoting technique was applied to a parallel corpus containing more than 106 million sentence pairs and over 2 billion English words, spanning 22 pivot languages. Word-level alignments inside the parallel sentences were done automatically, and thus do contain some noise; this results in paraphrase pairs of varying quality. In addition, PPDB paraphrases are not always synonyms, but are often categorized by other types of relationships, as in WordNet. To address these issues and promote good paraphrase pairs, the PPDB 2.0 score was introduced which reflects the strength of paraphrase relations (Pavlick et al., 2015). This work also classified the relation of each paraphrase pair (P_1, P_2) as one of six entailment relations:

- **Equivalence**, where P_1 and P_2 are true synonyms (e.g.,*distant* vs. *remote*)
- **Forward Entailment**, where P_1 is a specific type of P_2 (e.g., *mosquito* vs. *bug*)
- **Reverse Entailment**, where P_2 is a specific type of P_1 (e.g., *bug* vs. *mosquito*)
- **Exclusion**, where P_1 has the opposite meaning of P_2 (e.g., *nobody* vs. *someone*)
- **Independent**, where P_1 and P_2 have no relation (e.g., *car* vs. *family*)
- **Other Related**, where P_1 and P_2 have some relation other than entailment (e.g., *swim* vs. *water*)

In order to leverage a paraphrase resource for the task of text simplification, it is critical to know given a paraphrase pair (P_1, P_2), which phrase is simpler than the other. The Simple Paraphrase Database (SimplePPDB) was created for this purpose, which is a set of 4.5 million simplification rules extracted from PPDB (Pavlick and Callison-Burch, 2016). These rules come with both the predicted strength of the paraphrase relation (PPDB 2.0 score) and a simplification confidence score. This takes into account the strength of the paraphrase relation, and how well the right side of the rule simplifies the word on the left. For example, *perish* $\rightarrow$ *die* has a confidence score of 0.909, while *perish* $\rightarrow$ *murder* has a confidence score of 0.108. This simplification score was created by sampling 1,000

PPDB phrases, and extracting up to 10 of their paraphrases found in the resource. For each pair (P_1, P_2), crowdsourced annotations were collected which determine how well P_2 preserves the meaning of P_1, and which paraphrase is simpler than the other (or if there is no difference in complexity between the two). Pavlick and Callison-Burch (2016) train a multi-class logistic regression model on this data to predict if applying a paraphrase rule will result in a simpler or more complex output, or in an output that does not make sense. We show how we can effectively rank SimplePPDB paraphrases for an in-context lexical simplification task in Chapter 3.

2.3. Sentence Simplification Corpora

One of the main problems in simplification work is collecting a reasonable amount of quality data. Early works focused on specific aspects of simplification, allowing systems to leverage more general corpora. The lexical simplification work of Carroll et al. (1999) utilized lexico-semantic resources such as WordNet (Miller, 1992), while work focusing on deletion (Filippova and Strube, 2008) leveraged sentence compression corpora built from the British National Corpus and the American News Text Corpus. However, with the rise of statistical- and neural-based methods, which need to be trained on parallel sentences, collecting sentences containing a variety of simplification operations has become increasingly important.

2.3.1. Simple Wikipedia

In order to build an effective statistical-based system that takes into account all aspects of text simplification, Zhu et al. (2010) created a corpus of parallel sentences by leveraging Wikipedia and Simple English Wikipedia, which is tailored for younger children and adult English language learners. To do this, the authors first aligned regular and Simple English Wikipedia documents by tracking the language link found in Wikipedia dumps. After segmenting the articles into sentences, Zhu et al. (2010) aligned similar sentences in the aligned articles using sentence-level Term Frequency-Inverse Document Frequency (TF-IDF) (Nelken and Shieber, 2006). TF-IDF is a standard scoring scheme which indicates the importance of a particular word to a document within a corpus (Salton and Buckley, 1988). In order to compute sentence-level TF-IDF, Nelken and Shieber (2006) consider each sentence as a document. They then can define the weight w for a term t in a sentence s as follows:

$$w_s(t) = TF_s(t) * \frac{N}{DF(t)} \tag{2.1}$$

In this formula, $TF_s(t)$ represents whether or not the token t is found in sentence s, N is the total number of sentences in the corpus, and DF represents the number of sentences in which t is found.

The PWKP corpus has been widely used in simplification research, since it allows for the training of advanced statistical simplification systems (Woodsend and Lapata, 2011; Coster and Kauchak, 2011; Wubben et al., 2012). However, in their descriptive statistics of the dataset, Zhu et al. (2010) show that the aligned Simple English Wikipedia sentences are only slightly shorter than their corresponding Wikipedia sentences (20.87 vs. 25.01 tokens per sentence). Siddharthan (2014) further points out it is necessary to perform a more exhaustive examination of the quality of Simple English Wikipedia. The main reason is that, contrary to machine translation evaluation, which typically relies on native speakers to judge quality, it is generally difficult to find a "native Simple English speaker" because most adult English speakers are relatively advanced.

Taking this analysis further, Xu et al. (2015) perform an in-depth analysis of Simple English Wikipedia and PWKP. The authors extract 200 random sentence pairs from PWKP, and find that 50% of the pairs were either not correctly aligned (17%) or not good examples of actual simplifications (33%). Additionally, only 12% of the sentence pairs feature both deletion and paraphrasing, two key operations in simplification. Xu et al. (2015) argue that this is likely due to the fact that Simple English Wikipedia was created by volunteers, and these articles are also rarely complete rewrites of the original Wikipedia articles, often

Statistic	**L4**	**L3**	**L2**	**L1**	**L0**
# words/doc	1,152.01	996.59	931.78	799.48	676.20
# words/sent	23.23	19.44	16.60	14.11	11.91

Table 1: Descriptive statistics depicting the difference in complexity levels in the Newsela corpus (Xu et al., 2015). Here, **L3** is the least simplified version of the original document, while **L0** is the most simplified version.

leaving much of the information out.

2.3.2. The Newsela Corpus

To reduce the field's reliance on Simple Wikipedia, Xu et al. (2015) proposed a new corpus based on articles collected by Newsela, an education company that focuses on creating K-12 reading materials. This corpus (hereafter called Newsela V1) initially consisted of 1,130 news articles on a variety of topics, which were rewritten at four complexity levels by Newsela editors. In what follows, we describe the original document as Level 4 (L4), and the simplified documents as Level 3 (L3) to Level 0 (L0), depending on their complexity level. Table 1 shows descriptive statistics about the complexity of documents and sentences at the five Newsela levels. We can see that L0 documents indeed have on average significantly shorter sentences, a significant improvement over PWKP.

To extract parallel sentences, Xu et al. (2015) first align sentences from an article with complexity c with the most similar sentence in the same article with complexity $c - 1$. Unlike in PWKP, sentential similarity is calculated using the proportion of overlapping lemmatized words between two sentences s_1 and s_2:

$$sim(s_1, s_2) = \frac{|lemmas(s_1) \cap lemmas(s_2)|}{|lemmas(s_1) \cup lemmas(s_2)|} \tag{2.2}$$

In order to analyze the quality of the Newsela corpus, Xu et al. (2015) manually annotate 50 sentences from each level. This experiment finds that in L2, 34% of sentences are not simpler, while in L0, only 6% of the sentences are considered not simpler, and 68% had undergone both deletion and paraphrase operations. In total, this version of the Newsela corpus includes 141,582 sentence pairs.

Although the Newsela corpus was an improvement over previous datasets, the initial alignment algorithm used struggled to effectively align longer sentences, as these sentences were likely to have been changed significantly or split into multiple sentences in the simplified articles. Recently, Jiang et al. (2020) introduced a two-step process to better align sentences from parallel corpora. This work starts with an initial paragraph alignment step. Given two paragraphs in aligned documents $p_1 \in D_{i,L(j)}$ and $p_2 \in D_{i,L(j')}$, paragraph-level semantic similarity score ($simP$) is computed by taking an average of the maximum sentence-level similarities for each sentence $s_1 \in p_1$ and $s_2 \in p_2$). A pair of paragraphs (p_1, p_2) is considered aligned if they are both similar *and* found in similar positions in their respective documents; additionally, paragraphs are aligned if two continuous paragraphs $(p_2, p_3) \in D_{i,L(j')}$ are both relatively similar to p_1.

Next, Jiang et al. (2020) train a neural conditional random field (CRF) model to identify similar sentences within the aligned paragraphs. This model is able to leverage both sentence-level similarities as well as an alignment label transition, which accounts for the fact that if a complex sentence c_i is aligned with a simple sentence s_j, it's likely that c_{i-1} is aligned with s_{j-1}, and c_{i+1} is aligned with s_{j+1}. In this work, semantic similarity is computed by fine-tuning BERT (Devlin et al., 2019) on manually labeled data. Compared with the first version, these changes result in a significant increase in sentence pairs that contain both sentence splitting and additional rewrite operations, while also increasing the number of aligned sentences to 666,645. Note that this is partially due to the second version of the Newsela corpus (Newsela V2) being larger and containing 1,882 aligned articles, but the improvements introduced in this work further increase both the quality and quantity of parallel sentences.

2.4. Phrase-based and Statistical Text Simplification

Most recent work has attempted to solve the problem of text simplification holistically, formulating the task as a monolingual translation problem. Here, the input is a complex English sentence and the output is a sentence in simplified English. Training on a large corpus of aligned sentences allows models to perform transformation operations needed for simplification simultaneously. In this section, we discuss various statistical simplification systems, which have served as building blocks for later neural-based simplification models. We discuss in detail a seminal work on combining simplification subtasks into a holistic system (Zhu et al., 2010); a hybrid phrase-based model that has been proposed in order to better incorporate deletion and sentence splitting (Narayan and Gardent, 2014); and finally a statistical machine translation approach augmented with lexical substitutions (Xu et al., 2016).

While these methods have attempted to address the problem of sentence simplification holistically, in general these models are still broken down into several sub-models. This is not always a bad idea, particularly when phenomena related to simplification are not seen much in the training data, as is often the case for sentence splitting. However, under the assumption that the data is sufficient, a single model that jointly learns to perform all aspects of simplification simultaneously would likely generate more coherent and simpler output.

2.4.1. Tree-based Simplification Model

The first work that focused on integrating simplification subtasks into a holistic sentence-level simplification system is the Tree-based simplification model (TSM) (Zhu et al., 2010). This work integrates sentence splitting, deletion, reordering, and substitution operations into a single cohesive model. This work was also the first to effectively leverage Simple Wikipedia in order to collect parallel sentences.

The first step in TSM is to perform sentence splitting, if needed. To do this, Zhu et al. (2010)

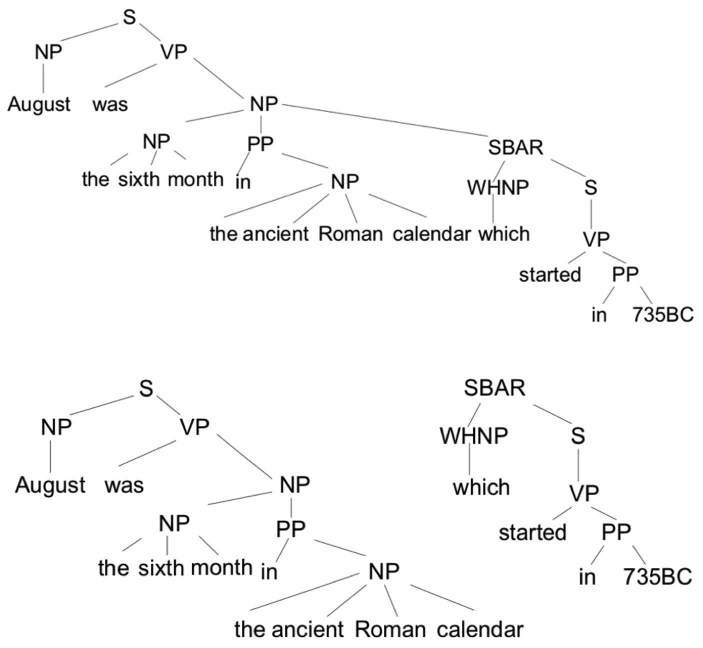


Figure 6: A potential split based on the parse tree of the sentence, "August was the sixth month in the ancient Roman calendar which started in 735BC.". Example recreated from Zhu et al. (2010).

decide whether or not to split a sentence on a boundary word, such as "which" in Figure 6, by using the relative length of the original sentence compared to the simplified version. Subsequently, the model determines whether to drop the boundary word by considering the word itself and the corresponding constituent within the parse tree; in Figure 6, this corresponds to "which" and "WHNP", respectively.

Similarly, for deletion, Zhu et al. (2010) utilize a word's direct parent node along with the constituent pattern of its children. In the example in Figure 6, the phrase "the sixth month" represents the constituent rule "NP $\rightarrow$ DT JJ NN". The model then probabilistically determines which parts of the rule should be kept, and which can be deleted.

Finally, for substitution, Zhu et al. (2010) use of a substitution table that tracks the probabilities $P(s|w)$, where w is a word and s is any candidate substitute for w. We can easily extend this to phrasal substitution by instead tracking the probabilities $P(s|n)$, where n is a non-terminal node in the constituency parse. The decision to make a phrasal or lexical substitution is determined by the node n having a higher probability than the most likely substitutions for all children of n. To calculate the probabilities in the substitution tables, these are initialized to a uniform distribution, and then updated by running an Expectation Maximization (EM) algorithm on the parallel sentences from PWKP.

Subsequent work built on the phrase-based machine translation framework by incorporating additional deletion (PBMT) (Coster and Kauchak, 2011) and post-hoc reranking (PBMT-R) (Wubben et al., 2012) mechanisms. In addition, Woodsend and Lapata (2011) presented a phrase-based model based on quasi-synchronous grammar to capture structural mismatches and complex rewrite operations.

2.4.2. Adding Deep Semantics to Phrase-based Machine Translation

While phrase-based systems such as TSM (Zhu et al., 2010) perform reasonably well when making simplifications at the lexical and phrasal level, they tend to perform poorly when deleting tokens and splitting sentences, two key aspects of simplification. To circumvent this problem, Narayan and Gardent (2014) incorporate better pre-processing steps for deletion and sentence splitting, before utilizing a probabilistic phrase-based system. Regarding sentence splitting, the authors argue that a purely syntactic approach (Zhu et al., 2010) is often inadequate. This is because in cases where the split sentences share a phrase, relying on syntax alone often fails to correctly generate this phrase in both sentences. Instead, the authors recommend using a formal semantic representation of the sentence which is derived from Discourse Representation Theory (DRT). In particular, the Hybrid model takes as input a Discourse Representation Structure (DRS). A simple example sentence and its corresponding DRS is shown below:

- **Sentence**: The man walks the dog.
- **DRS**: [x,y: man(x), dog(y), walks(x, y)]

With this representation, it is possible to clearly identify phrases that are shared across different parts of the sentence, and which are critical components, e.g. agents or patients.

Similarly, the authors leverages the DRS to determine what parts of the sentence should be deleted. The model can determine which phrases are crucial components, and which are optional content that can potentially be removed. In contrast, work that only considers the input sentence or the syntactic structure does not discriminate between phrases, and thus can delete important parts of the sentence.

The last component is to further simplify by making lexical and phrasal substitutions, and reordering the sentence. Narayan and Gardent (2014) train a standard phrase-based machine translation system on PWKP. In addition, to ensure the output is relatively fluent, the authors train a general language model on Simple Wikipedia, and integrate this into their phrase-based simplification system. Altogether, the Hybrid model can be defined as follows:

$$Hybrid(c) = \underset{s}{\operatorname{argmax}}\; P(DRS_c|s')P(s'|s)P(s) \tag{2.3}$$

Here, c represents the original complex sentence, DRS_c is the semantic representation of c, s' represents the intermediate sentence(s) after performing splitting and deletion, and s is a candidate simplified sentence. Their results show that their hybrid method greatly outperforms previous phrase-based simplification systems in terms of simplicity, while remaining competitive on fluency and adequacy.

2.4.3. Integrating Lexical Simplification into Statistical Machine Translation

In contrast to previous phrase-based work which utilizes several components to model different aspects of simplification, Xu et al. (2016) instead start from a standard statistical machine translation approach and integrate several simplification-specific aspects, focusing mainly on substitution. In addition, for substitution this work leverages the Paraphrase database (PPDB), a collection of more than 100 million English paraphrase pairs (Ganitkevitch et al., 2013). While PPDB rules are not simplification-specific, the sheer size allows for much greater coverage compared to previous simplification datasets, and non-simplifying rules can subsequently be filtered out.

To properly leverage PPDB rules for simplification, Xu et al. (2016) train a linear model $l_{\mathbf{w}}(c, s)$, where c is the original complex sentence, s is the candidate simplified sentence, and $\mathbf{w}$ are the weights learned by the model. As features, the authors first consider all features available in PPDB for each paraphrase pair (p_1, p_2); these include the conditional paraphrase probability $P(p_2|p_1)$, the distributional similarity between the paraphrases' contexts, part-of-speech information, n-gram based features for words seen in the left and right of a phrase, among others. Beyond paraphrase-specific features, Xu et al. (2016) also calculate paraphrase length in characters, their length in words, the maximum number of syllables, and the fraction of common English words included in the paraphrases. These features are computed for both p_1 and p_2.

In order to learn $\mathbf{w}$, Xu et al. (2016) use a pairwise ranking optimization algorithm, which differentiates a good simplification s_g from a bad simplification s_b using some automatic metric m. In other words, the pair (s_1, s_2) is labeled as 1 if $m(c, s_1) > m(c, s_2)$, and as 0 otherwise, for some metric m. This work tries three metrics as m: BLEU, a standard n-gram-based metric to measure quality in machine translation (Papineni et al., 2002); FKBLEU, a novel metric which combines BLEU with the difference in Flesch-Kincaid Grade Level between the original sentence and the candidate (Kincaid et al., 1975); and SARI, a second novel metric which tracks how many n-grams are correctly kept, added, and deleted by the candidate. These metrics are discussed in detail in Section 2.6.

Using both automatic metrics as well as human judgments of fluency, adequacy, and number of good simplifications made (simplicity+), Xu et al. (2016) find that optimizing on BLEU results in candidates that are identical to the original sentence; this is one of many experiments that demonstrates the fact that BLEU is generally not a good metric for optimizing simplification systems (Sulem et al., 2018). The combination of PPDB and SARI seems to perform best overall, making more simplifications while preserving fluency and adequacy better than the previous state-of-the-art system of Wubben et al. (2012). However, the number of actual simplifications is still quite low compared to the reference sentences (0.65 vs 1.35).

2.5. Neural Machine Translation and Applications to Simplification

Sequence-to-Sequence (Seq2Seq) models, which learn mappings from one sequence to another using a neural network, have had enormous success in not just text simplification, but also a variety of natural language processing applications, notably including machine translation (Sutskever et al., 2014; Luong et al., 2015; Vaswani et al., 2017), text summarization (Nallapati et al., 2016), and dialog systems (Vinyals and Le, 2015). In this section, we first present an overview of the Seq2Seq framework, followed by a discussion of recent applications of Seq2Seq models to text simplification. We also include a description of common decoding strategies.

2.5.1. Sequence-to-Sequence Models

Seq2Seq models generally include an encoder and a decoder model. The encoder transforms some input $\mathbf{x}$ into a fixed-size latent representation, $z \in \mathbb{R}^V$, while the decoder transforms this representation to output a conditional probability for each word in the target sequence, $\mathbf{y}$, given the input sequence and tokens generated so far. Here, V is the cardinality of the enumerated vocabulary $\mathcal{V}$. Generally, the encoder and decoder are both Recurrent Neural Networks (RNNs) (Sutskever et al., 2014), though more recent works have sometimes replaced these with attention-based Transformer networks (Vaswani et al., 2017). The encoder and decoder are then jointly trained in order to maximize the conditional probability of the target sentence given the source sentence, i.e. $P(\mathbf{y}|\mathbf{x})$.

At every step t of the input sequence, the encoder generates the hidden state as follows:

$$h_t = f(x_t, h_{t-1}) \tag{2.4}$$

where x_t is the current token in the sentence and h_{t-1} is the hidden state at the previous time step. Here, the function $f(.)$ is generally a Long Short-Term Memory (LSTM) network. The final hidden state of the encoder is then taken as the context vector c to be passed to the decoder.

At each time step t, the decoder generates the next token y_t and its hidden state d_t as follows:

$$d_t = f(d_{t-1}, y_{t-1}, c)$$

$$P(y_t | y_{1:t-1}, c) = g(d_t, y_{t-1}, c)$$

Here, the function $g(.)$ is a softmax layer to convert scores into a valid probability distribution.

During the decoding process, most strategies attempt to find the most likely overall sequence, i.e. choose a $\hat{\mathbf{y}}$ such that:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \prod_{t=1}^{N} P(y_t \mid y_{1:t}, \mathbf{x})$$

However, unlike Markovian processes, no sub-exponential algorithm exists to find the best decoded sequence, and thus we instead use approximation algorithms. The simplest approach to decoding a likely sequence is to greedily select the most likely word at each timestep:

$$\hat{y}_t = \arg\max_{y_t} P(y_t | y_{1:t-1}, \mathbf{x})$$

There are several key limitations in this original architecture. Since the last step of the encoder is the only thing that is passed to the decoder, it is expected to contain information from the entire sentence; this problem, known as bottlenecking, becomes increasingly problematic with longer sentences. In addition, recurrent neural networks (RNNs) that were generally used as the encoder and decoder models are generally somewhat slow to train; since each RNN step takes the output from the previous step, only one step can be computed at a time.

To address the bottlenecking problem, the concept of *attention* was introduced, which allows the decoder to focus on the relevant parts of the source sentence (Bahdanau et al., 2015; Luong et al., 2015). With this approach, the context vector at time step t, c_t, is no longer static; instead, it now depends on the entire *sequence* of hidden states.

$$c_t = \sum_{t=1}^{T_{\mathbf{x}}} \alpha_{it} h_t$$

In this formula, $T_{\mathbf{x}}$ represents the number of time steps in the input $\mathbf{x}$, and α_{it} represents the weight of each hidden state. The decoder now receives all encoder hidden states, each of which represents the addition of a single token from the source sentence. Each step in the decoder learns a score for each encoder state, where important states are given higher scores. Finally, we take the softmax of these scores, and multiply them by the hidden states to get an overall context vector. Incorporating attention into Seq2Seq models was shown to greatly improve performance on machine translation and other generation tasks.

To speed up training, Vaswani et al. (2017) proposed removing the RNN component of the model altogether and to rely solely on attention; this is known as the *transformer* architecture. The model still consists of separate encoder and decoder components. In the original architecture, the encoder is six encoders stacked on top of each other; the decoder is set up in the same way. Each encoder contains two layers: a self-attention layer, followed by a feed-forward neural network. For each word w, the self-attention layer looks at the rest of the input to better encode w. Since the attention mechanisms has no knowledge of the position of each word in the input, before being passed into the encoder, the embeddings corresponding to the input text are modified using positional encodings. The decoding side is similarly structured, with the output from the previous decoder layer being used as input to the subsequent layer. After the decoders, there is a final linear layer, followed by a softmax layer to produce a final probability distribution over the vocabulary. Following the standard Seq2Seq framework, after each word is predicted, the output sequence so far is passed through the decoder as additional context.[1] The transformer architecture is much faster to train and also delivers better results when compared to previous state-of-the-art RNN-based Seq2Seq models.

2.5.2. Applications of Seq2Seq models to Text Simplification

After the success of Seq2Seq models in machine translation and other text generation tasks, it was natural for this architecture to be applied to text simplification as well, as this has often been viewed as a monolingual machine translation task from complex to simple English. Nisioi et al. (2017) proposed the first major application of Seq2Seq models to text simplification, applying the standard encoder-decoder approach with attention, and using beam search as their decoding strategy. This work shows that a vanilla Seq2Seq model directly applied to text simplification is able to generate output that is more grammatical and better at preserving the meaning of the original sentence compared to previous statistical and phrase-based models.

However, although a higher proportion of the changes proposed by Seq2Seq models are high quality, they still make significantly fewer changes compared to previous approaches (Nisioi et al., 2017). This illustrates one of the key challenges in applying standard Seq2Seq models to simplification. If we consider sentence pairs in the Newsela dataset, 73% of tokens are copied from the original to the simplified version in Newsela, making the copy operation by far the most common operation (Zhang and Lapata, 2017). Because standard Seq2Seq models are extremely good at picking up on patterns present in the data, they naturally learn to copy very well, resulting in output that is often either identical to the original sentence, or only with a few small changes. Due to this issue, most subsequent work focuses on how to encourage the model to make more simplification operations that do not involve copying from the original.

[1]This description of transformers is partially based on the blog post http://jalammar.github.io/illustrated-transformer/.

To counteract this tendency of the models, Zhang and Lapata (2017) integrate reinforcement learning into the Seq2Seq framework. This rewards the model for producing output that is simpler than the original text but also is grammatical (fluency) and perserves the meaning of the original sentence (adequacy). To do this, we can view the standard Seq2Seq model as the agent who generates an output sequence $\hat{\mathbf{y}}$ according to its current "policy", defined in Equation 2.5.1. The output sequence, the original sentence $\mathbf{x}$, and the reference simplification $\mathbf{y}$ are then passed into several functions which generate a reward for that sequence, which is used by the REINFORCE algorithm to update the original agent.

Zhang and Lapata (2017) make use of three separate reward functions that score the quality of each generated output in terms of its fluency, adequacy, and simplicity. The fluency reward function is based on a language model trained on simple text, to simulate the likelihood that the generated output is a simple sentence. Formally, this function takes as input $\hat{\mathbf{y}}$, and calculates the normalized probability as follows (Zhang and Lapata, 2017):

$$r_F(\hat{\mathbf{y}}) = \exp\left(\frac{1}{|\hat{\mathbf{y}}|}\sum_{i=1}^{|\hat{\mathbf{y}}|} log(P_{LM}(\hat{y}_i|\hat{y}_{1:i-1}))\right) \tag{2.5}$$

In this formula, $P_LM(\cdot)$ represents the conditional probability assigned by the language model to the token generated at time t, given the previous tokens. Regarding adequacy, the reward function makes use of a sentential autoencoder (Dai and Le, 2015) trained jointly on the complex and simple sentences. In order to calculate how well the meaning of the original sentence is preserved, this function takes in $\mathbf{x}$ and $\hat{\mathbf{y}}$, converts them to vector representations $\mathbf{e_x}$ and $\mathbf{e_{\hat{y}}}$, and simply calculates their cosine similarity:

$$r_A(\mathbf{x}, \hat{\mathbf{y}}) = \mathbf{cos}\left(\frac{\mathbf{e_x} \cdot \mathbf{e_{\hat{y}}}}{||\mathbf{e_x}||\ ||\mathbf{e_{\hat{y}}}||}\right) \tag{2.6}$$

Finally, for simplicity the reward function utilizes SARI, an automatic metric commonly used to evaluate simplification systems (discussed more in Section 2.6.3) (Xu et al., 2016), which calculates the quality of $\hat{\mathbf{y}}$ given both $\mathbf{x}$ and $\mathbf{y}$. While this is the current metric that correlates best with simplicity, Zhang and Lapata (2017) argues that due to the noise in the corpora used for training (Zhu et al., 2010; Xu et al., 2015), it is important to also take into account "reverse SARI", which instead calculates the quality of $\mathbf{y}$ given $\mathbf{x}$ and $\hat{\mathbf{y}}$. Formally, the simplicity reward function is defined as follows; here, β is a hyperparameter to be tuned while training:

$$r_S(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) = \beta SARI(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) + (1 - \beta) SARI(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{y}) \tag{2.7}$$

Vu et al. (2018) extend the standard Seq2Seq framework to incorporate memory augmentation which simultaneously performs lexical and syntactic simplification, outperforming previous vanilla Seq2Seq models on both human and automatic metrics. Similarly, Zhao et al. (2018) propose DMASS (Deep Memory Augmented Sentence Simplification), a Transformer-based approach (Vaswani et al., 2017) which also integrates simplification rules. This work has also been shown to outperform previous models using automatic metrics. However, Zhao et al. (2018) critically do not perform a human evaluation; restricting evaluation to automatic metrics is generally insufficient for comparing simplification models. In Section 4.2.2, we gather human annotations to evaluate outputs from DMASS compared with other state-of-the-art simplification models, and find that DMASS was rated lower on fluency, adequacy, and simplicity. This is initially somewhat surprising, because Transformer architectures have generally led to large improvements on many NLP tasks (Vaswani et al., 2017). However, it is important to note that Transformers are extremely sensitive to hyper-parameter tuning (Popel and Bojar, 2018); thus, if not properly trained, these models will likely generate sub-optimal output. Indeed, subsequent attempts to train a transformer-based model were able to achieve far superior performance (Mallinson and Lapata, 2019b; Jiang et al., 2020) as estimated by human judges. The improvement seen in Jiang et al. (2020) is likely also due to initializing a Transformer with BERT embeddings (Devlin et al., 2019).

2.5.3. Decoding Strategies

As discussed above, greedy search is the simplest decoding strategy, and can be used if only one output sequence is required. However, greedy search is a deterministic approach, which typically yields repetitive and short output sequences. Additionally, it does not allow for generating multiple samples. Thus, in practice it is rarely used with modern language models.

Instead, the most common decoding strategies are random sampling and beam search. Random sampling, as the name implies, involves randomly sampling from the model's distribution at every time step. This allows for more uncommon words (according to language model probability) to sometimes be chosen, often allowing for more interesting output. In addition, the randomness involved in the process allows a user to quickly generate multiple non-identical candidate outputs. This can be extremely useful for more open-ended tasks such as conversational dialogue systems (i.e. chatbots) where multiple unique outputs can be valid.

Beam search approximates finding the most likely sequence by performing breadth-first search over a restricted search space. At every decoding step, the method keeps track of b partial hypotheses. The next set of partial hypotheses is chosen by expanding every path from the existing set of b hypotheses, and then choosing the b with the highest scores. Most commonly, the log-likelihood of the partial sequence is used as the scoring function. We present the standard beam search algorithm in Algorithm 1. Note that SOS represents the start of sentence token, and EOS the end of sentence token.

We discuss additional variations of these decoding strategies and compare their relative effectiveness in generating high-quality and diverse outputs in Chapter 5.

Algorithm 1 Beam Search Inference

```
1: procedure BEAM SEARCH
2:     $B \leftarrow \{SOS\}$
3:     $k \leftarrow$ BeamWidth
4:     $out \leftarrow k$-best output list
5:     while $|out| < k$ do
6:         $front \leftarrow$ remove all nodes from $B$
7:         for $w \in front$ do
8:             $succ \leftarrow w$'s $k$-best successors
9:             for $s \in succ$ do
10:                if $s == EOS$ then
11:                    $out \leftarrow out \cup \{s\}$
12:                else
13:                    $B \leftarrow B \cup \{s\}$
14:        Sort $B$
15:        if $|B| > k$ then
16:            Prune $B$ to $k$-best successors
       return out
```

2.5.4. Diversity Promotion During Training

In Section 5, we compare a variety of post-training diversity-promoting algorithms. Here, we discuss other related works that instead promote diversity at training time. Several works have attempted to encourage diversity during training by replacing the standard log-likelihood loss with a diversity-promoting objective. Li et al. (2016a) introduce an objective that maximizes mutual information between the source and target sequences. Zhang et al. (2018) use an adversarial information maximization approach to encourage generated text to be simultaneously informative and diverse. Xu et al. (2018) also use an adversarial loss; their loss function rewards fluent text and penalizes repetitive text. In our comparison of decoding strategies in Chapter 5, we do not use these methods, as they tend to be task-specific and difficult to implement. All of the diversity strategies we apply in this thesis share the trait that they are agnostic to the model architecture and the data type of the input, as long as the output of the model is a probability distribution over tokens in a sequence.

2.6. Evaluation of Simplification Systems

In many text generation tasks, it is important to evaluate model quality using human judgments (Bojar et al., 2016). However, this procedure is expensive, time consuming, and untenable when testing many model variations. Thus, there have been several attempts to develop automatic metrics for evaluating the quality of the generated texts. In this section, we first discuss human evaluation, before describing early automatic evaluation approaches and the current state-of-the-art metrics. At the end of the section, we discuss limitations that have not yet been addressed in simplification evaluation.

2.6.1. Human Evaluation

Despite the recent advances in automatic metrics, the most reliable method to compare simplification systems is still through the collection of human judgments. Given a complex sentence c and its simplified version s, judgments are typically collected along three dimensions, using either a three-point or a five-point Likert scale.

- **Fluency**: Is s grammatical, i.e. is s a well-formed sentence?
- **Adequacy**: Does s preserve the meaning of c?
- **Simplicity**: Is s simpler than c?

Simplicity is generally the most difficult dimension to evaluate, as it is the most open ended. Some work has attempted to make this dimension more quantifiable by asking workers to count the number of rewrites found in the simper sentence (Xu et al., 2015), however most recent papers have simply left the term "simpler" up to the interpretation of the workers (Zhang and Lapata, 2017; Jiang et al., 2020).

2.6.2. BLEU and FKGL

One of the earliest metrics for simplification is Flesch-Kincaid Grade Level (FKGL), which measures the readability of a text using surface-level features (Kincaid et al., 1975). Formally, FK is defined as follows:

$$FK = 0.39 \times \left(\frac{\#words}{\#sentences} \right) + 11.8 \times \left(\frac{\#syllables}{\#words} \right) - 15.59 \tag{2.8}$$

To determine the coefficients in Equation 2.8, Kincaid et al. (1975) used reading comprehension test scores on Navy personnel reading training manuals to train a regression.

Simplification systems are often inspired by machine translation models, as discussed in Sections 2.4 and 2.5. We also find a strong influence of translation evaluation practices in the simplification literature. The most commonly-used automatic metric is Bilingual Evaluation Understudy, or BLEU (Papineni et al., 2002). At the sentence level, BLEU compares a candidate translation to one or more reference translations by counting the number of overlapping n-grams. BLEU generally correlates well with human judgments of translation quality (Papineni et al., 2002; Doddington, 2002), though this does not hold when there are many translations of similar quality or phrasal permutation (Callison-Burch et al., 2006). Still, BLEU is reasonably intuitive and fast to use. This has led to its adaptation for the evaluation of a variety of monolingual text generation tasks, including summarization (Graham, 2015) and text simplification (Zhu et al., 2010; Woodsend and Lapata, 2011; Wubben et al., 2012).

While BLEU and FKGL are reasonable automatic metrics, they both have clear flaws when applied to modern text simplifications systems. Although FKGL is widely accepted for measuring readability, it relies on the assumption that the evaluated text is well-formed (Xu et al., 2016); but, in practice, generated simplifications often contain grammatical errors. In addition, a lower predicted readability level is insufficient for determining whether a candidate simplification is appropriate for a given sentence. BLEU would in theory be more appropriate as a holistic evaluation metric, because it compares to reference simplifications. However, in practice BLEU often falls short, correlating poorly with human judgments for lexical simplicity (Xu et al., 2016) and sentence splitting (Sulem et al., 2018), two critical components of the text simplification process. In addition, Sulem et al. (2018) show that BLEU often correlates negatively with human judgments on general simplicity due to BLEU's brevity penality, leading to the incorrect scoring of shorter and simpler sentences.

2.6.3. The SARI Metric

Given the flaws of previous metrics, Xu et al. (2016) argue that a quality simplification metric needs to not only take into account reference simplifications, but also the original sentence. The main reason is that a common operation in simplification is to copy words directly from the source sentence, which is uncommon in machine translation outside of named entities or untranslatable words. To effectively incorporate information from the original sentence, Xu et al. (2016) introduce two novel metrics: FKBLEU and SARI.

FKBLEU, as its name indicates, combines the readability aspects captured by FKGL with the overall appropriateness captured by a modified version of BLEU, known as input-aware BLEU (iBLEU). iBLEU extends BLEU to take into account the input sentence, which allows the metric to measure the diversity of paraphrase outputs, as well as the quality of these paraphrases (Sun and Zhou, 2012). Taking into account this diversity avoids rewarding the model too much for simply copying directly from the input sentence. Formally, iBLEU for simplification is defined as follows:

$$iBLEU(C, \mathbf{R}, S) = \alpha BLEU(S, \mathbf{R}) - (1-\alpha) BLEU(C, S) \tag{2.9}$$

In Equation 2.9, C represents the original complex sentence, $\mathbf{R}$ represents the reference simplifications, and S represents the candidate simplification generated by the model. α is a hyperparameter that determines how much weight should be given to the adequacy and dissimilarity aspects, and is generally set to 0.9 (Sun and Zhou, 2012).

To create FKBLEU, Xu et al. (2016) combine the geometric mean between iBLEU and the difference in readability between the original sentence and the candidate simplification, as measured by FKGL. We define FKBLEU formally below (note that we use the same variables from Equation 2.9):

$$FKBLEU(C, \mathbf{R}, S) = iBLEU(C, \mathbf{R}, S) \times FKDiff(C, S) \tag{2.10}$$

$$FKDiff(C, S) = Sigmoid(FKGL(C) - FKGL(S)) \tag{2.11}$$

Algorithm 2 Evaluating Addition Operation

```
 1: procedure Recall and Precision
 2:     for $n \in [1, 4]$ do
 3:         $correct \leftarrow 0$, $total_p \leftarrow 0$, $total_r \leftarrow 0$
 4:         for $t \in S_n$ do
 5:             if $t \in S_n \cap \mathbf{R}_n \cap \overline{C}_n$ then
 6:                 $correct$ += 1
 7:             if $t \in S_n \cap \overline{C}_n$ then
 8:                 $total_p$ += 1
 9:             if $t \in \mathbf{R}_n \cap \overline{C}_n$ then
10:                 $total_r \leftarrow total_r + 1$
11:             for $r_n \in \mathbf{R}_n$ do
12:                 $r_n \leftarrow r_n - \{t\}$
13:         $precision_n = \frac{correct}{total_p}$
14:         $recall_n = \frac{correct}{total_r}$
15:     $precision \leftarrow \frac{1}{4} \sum_{n \in [1,4]} precision_n$
16:     $recall \leftarrow \frac{1}{4} \sum_{n \in [1,4]} recall_n$
17:     $F\text{-}score \leftarrow 2 \times \frac{precision \times recall}{precision + recall}$
        return $F\text{-}score$
```

Xu et al. (2016) also introduce SARI, a novel metric for evaluating the quality of simplification systems' output at the lexical level. Similar to FKBLEU, SARI utilizes both the original sentence along with reference sentence(s). Specifically, SARI breaks down simplification evaluation into three key aspects: how often the generated sentence S correctly preserves tokens from the original sentence C (keep), according to the reference sentences $\mathbf{R}$; how often S correctly adds tokens to C (addition); and how often S correctly deletes tokens from C (deletion).

To determine how to calculate the F-score for each individual operation, let's consider the

Metric	References	Fluency	Adequacy	Simplicity+
FKGL	none	-0.002	0.136	0.147
BLEU	single	0.366	0.459	0.151
BLEU	multiple	**0.589**	**0.701**	0.111
FKBLEU	multiple	0.349	0.410	0.235
SARI	multiple	0.342	0.397	. **0.343**

Table 2: Comparison of various metrics with human judgments on fluency, adequacy, and simplicity gain (Xu et al., 2016).

addition operation, as an example. In this section, we will denote S_n as all n-grams from a sentence S. Intuitively, this operation tracks three main aspects: additions made in the generated simplification ($S_n \cap \overline{C}_n$); n-grams in the reference simplifications not found in the original complex sentence ($\mathbf{R}_n \cap \overline{C}_n$); and the number of additions made that were also found in at least one of the reference sentences ($S_n \cap R_n \cap \overline{C}_n$). Overall precision and recall is then computed by averaging 1-gram, 2-gram, 3-gram and 4-gram precision and recall, and F-score is then reported as the final result. The full description of how to calculate the F-score for the addition operation is given in Algorithm 2. The process for calculating the F-scores for the deletion and keep metrics follows a similar logic.

To show that their metrics better capture differences in simplification quality than previous metrics, Xu et al. (2016) generate outputs from several state-of-the-art systems. They then collect human judgments of fluency and adequacy. In addition, the authors consider a modified simplification measure called simplification gain, which asks annotators to count the number of good lexical and/or syntactic simplifications that were made in the generated sentences. This approach helps annotators focus specifically on lexical and phrasal simplification, resulting in higher inter-annotator agreement.

As shown in Table 2, BLEU correlates strongly with judgments on fluency and adequacy, but poorly with simplicity estimates. On the other hand, SARI correlates best with simplicity judgments, while still correlating reasonably well with fluency and adequacy (though worse than BLEU). Xu et al. (2016) discusse this tradeoff in detail. The authors point out that BLEU generally scores a sentence highly when there are relatively few changes from the original sentence; this is because with multiple reference sentences, it is very likely that most n-grams from the original sentence will be found in at least one of the references. Similarly, human annotators will trivially rate these sentences high with regards to adequacy; in addition, since the original sentences are human-generated, they also tend to be grammatical. However, several candidates containing little or no changes from the original sentence would naturally result in lower simplicity judgments, thus SARI is more appropriate for estimating the overall quality of simplification systems than previous automatic metrics. However, as SARI only shows mild correlation with all three simplification dimensions, collecting human judgments is still viewed as the most reliable way to compare sentence simplification systems.

2.6.4. Quality Estimation

Quality Estimation (QE) methods were first introduced in the field of machine translation to measure the quality of automatically translated text without need for reference translations (Bojar et al., 2017; Martins et al., 2017; Specia et al., 2018). Recently, a QE approach has been proposed for summarization evaluation. Xenouleas et al. (2019) propose several extensions to the BERT fine-tuning process (Devlin et al., 2019) to estimate summary quality. Their proposed model, Sum-QE, predicts five linguistic qualities of generated summaries using multi-task training: Grammaticality, Non-redundancy, Referential Clarity, Focus, and Structure and Coherence. Apart from the state-of-the-art results obtained by BERT on many classification tasks, Xenouleas et al. (2019) show that it can also be successfully applied to QE. In Section 4.3, we adapt Sum-QE to simplification QE, and estimate the Fluency, Adequacy and Complexity of simplified text.

Alva-Manchego et al. (2019) recently created a toolkit to calculate various standard automatic simplification metrics, including SARI, word-level accuracy scores, and QE features such as the compression ratio and the average number of added/deleted words. Recent works that addresses QE for simplification have experimented with a variety of features, including sentence length, average token length, and language model probabilities (Stajner et al., 2016; Martin et al., 2018). However, the best models from these works also use reference-reliant features such as BLEU and translation error rate, as these have been shown to correlate well with fluency and adequacy. Note that these works were carried out before the rise of large-scale pre-trained models (Peters et al., 2018; Devlin et al., 2019).

2.7. Large-Scale Pre-trained Language Models

The advancements in attention-based Sequence-to-Sequence translation models had broad implications in other NLP tasks. In particular, they allowed for the creation of large-scale pre-trained language models, including ELMO (Peters et al., 2018), GPT (Radford and Narasimhan, 2018), ULMFiT (Howard and Ruder, 2018), and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), which we make extensive use of in this thesis. BERT, like the Transformer, is a set of self-attention-based encoders stacked on top of each other. These encoders are trained on two tasks. The first is a masked language modeling task, where 15% of the words in the input are randomly selected to be removed, and the model needs to predict these words. The second is a next sentence prediction task, where given two sentences, the model needs to determine if the second sentence is indeed the one that should follow directly after the first. The result of this general pre-training is a model that can generate word-level embeddings that take into account the context surrounding each word. While these embeddings by themselves are already quite useful for tasks such as predicting textual similarity, the most important property of this model is that it can be fine-tuned for a variety of tasks simply by adding a single output layer on top of the original architecture, and training on a relatively small amount of task-specific data. This has shown to produce state-of-the-art results in many NLP tasks, including sentiment analysis, question answering, and common-sense inference, among many others (Devlin et al., 2019).

Since BERT was introduced, there have been many follow-up works that have attempted to further improve upon this model. These extensions include: optimizing BERT's training and removing the next sentence prediction task (Liu et al., 2019); incorporating parameter-reduction techniques to lower memory usage and increase training speed (Sanh et al., 2019; Lan et al., 2020); relying on an autoregressive framework, which circumvents the need for using artificial [MASK] symbols (Yang et al., 2019); and changing the pre-training objective to a discriminator (Clark et al., 2020).

Traditionally, combining word-level representations using a simple mean pooling strategy has been shown to be surprisingly effective for measuring textual similarity (Faruqui et al., 2015; Yu et al., 2014; Tom Kenter and de Rijke, 2015). Combining subword-level BERT embeddings in a similar way does a reasonable job of ranking texts by their similarity; however, doing so results in embeddings that are extremely close together, making it difficult to compare embedding similarity using the traditional cosine similarity measure. To alleviate this issue, SBERT fine-tunes BERT on Natural Language Inference data to create sentence embeddings that are more semantically meaningful (Reimers and Gurevych, 2019). Specifically, SBERT relies on a Siamese BERT-network architecture, which involves passing two text segments (T_1, T_2) through two BERT networks that share the same weights. The model performs mean pooling on the output word-level embeddings, resulting in sentence embeddings (e_{T_1}, e_{T_2}). These embeddings are concatenated together, along with their absolute difference $|e_{T_1} - e_{T_2}|$, before being passed through a final Softmax layer to predict the label of (T_1, T_2); for the NLI task, the labels are *entailment*, *contradiction*, and *neutral*.

CHAPTER 3 : Simplification Using Paraphrases and Context-based Lexical Substitution

3.1. Introduction

Early text simplification works break down the problem into sub-tasks, to make it more tractable. These sub-tasks include complex word and sentence identification, lexical simplification, syntactic simplification, and sentence splitting (Saggion, 2017). In this section, we focus on lexical simplification, the task of replacing difficult words in a text with words that are easier to understand. Lexical simplification can involve identifying complex words in context (Shardlow, 2013b), context-aware lexical substitution (Biran et al., 2011), and finally choosing the appropriate simple word. This last step is often done by ranking words based on their frequency in a large general corpus (Devlin and Tait, 1998).

To identify complex words, we train a model on data manually annotated for complexity. Unlike previous work, our classifier takes into account both lexical and context features. We extract candidate substitutes for the identified complex words from SimplePPDB (Pavlick and Callison-Burch, 2016), a database of 4.5 million English simplification rules linking English complex words to simpler paraphrases. We select the substitutes that best fit each context using a word embedding-based lexical substitution model (Melamud et al., 2015). We show that our complex word identification and substitution model improves over several baselines which exploit other types of information and do not account for context. Our approach proposes accurate substitutes that are simpler than the target words and preserve the meaning of the corresponding sentences.

3.2. Complex Word Identification

The first step for lexical simplification is to identify the complex words that should be simplified. To accomplish this task, Shardlow (2013b) proposed several features that help to determine whether a word is complex, including word length, number of syllables, word frequency, number of unique WordNet synsets and synonyms. We also leverage these features in this work. For word frequency, Shardlow (2013b) used frequency counts collected from SUBTLEX, a corpus of 51 million words extracted from English subtitles.[1]. In contrast, we use word-level frequencies from the Google Web1T corpus (Brants and Franz, 2006) (henceforth Google n-gram), as this is significantly larger and contains texts across many genres and time periods, making it more suitable for extracting general word frequency information.

We add to these features from previous workbinary Part-of-Speech (POS) features and Word2Vec embeddings (Mikolov et al., 2013a). Embeddings in particular are important as they can serve to capture the context in which a word is typically found, and a word's complexity can often be influenced by its context. Following this intuition, we additionally include several explicit context-based features: average length of words in the sentence, average number of syllables, average word frequency, average number of WordNet synsets and synonyms, and sentence length. We train Support Vector Machine (SVM) classifiers using these features.

The experiments from our initial work on complex word identification occurred prior to the rise of large-scale pre-trained language models. Thus, as a sanity check, we now present two additional models that show how recent improvements in NLP can help with our task. Specifically, we train two models. First, we fine-tune RoBERTa (Liu et al., 2019) using only a single word and a binary label indicating complexity. We expect that this model would not work that well, because the main idea of BERT-based embeddings is that they are *contextual*, but this initial approach does not take advantage of this. Thus, in order to more effectively leverage contextual embeddings, we also fine-tune RoBERTa with both the word and the sentence it originally came from concatenated together. Specifically, the input to this model looks as follows:

[1]SUBTLEX can be found at:
https://www.ugent.be/pp/experimentele-psychologie/en/research/documents/subtlexus

	Annotators	Prevalence	Example Words
least complex	0	0.617	heard, sat, feet, shops, town
	1	0.118	protests, pump, trial
	2	0.062	sentenced, fraction, primary
	3	0.047	measures, involved, elite
	4	0.035	fore, pact, collapsed
	5	0.031	slew, enrolled, widespread
	6	0.029	edible, seize, dwindled
	7	0.023	perilous, activist, remorse
	8	0.023	vintners, adherents, amassed
most complex	9	0.015	abdicate, detained, liaison

Table 3: Examples of words identified as complex (i.e. difficult to understand within a text) by n annotators, where $0 \leq n \leq 9$. Column 2 (Prevalence) shows the proportion of the total number of words identified as complex by n annotators.

[CLS] <*word*> [SEP] <*sentence-containing-word*> [SEP]

3.2.1. Complex Word Identification Data

We present a new corpus with complexity annotations for training and testing our models. We hired annotators on Amazon Mechanical Turk; their task consisted of identifying complex words in the context of given texts. We randomly selected 200 texts from the Newsela corpus (Xu et al., 2015), and had the first 200 tokens from each text labeled by nine annotators. We pre-process the texts using the Stanford CoreNLP suite (Manning et al., 2014) for tokenization, lemmatization, POS tagging, and named entity recognition. The annotators were instructed to label at least 10 complex words they deemed worth simplifying for young children, people with disabilities, and second language learners. After filtering out stop words (articles, conjunctions, prepositions, pronouns) and named entities, we are left with 17,318 labeled tokens. For our binary classification task, tokens labeled by at least three annotators are considered to be complex, and tokens labeled by less than three annotators are considered to be "simple". This increases the likelihood of complex segments being actually complex; as we can see from Table 3, tokens identified by only one or two annotators tend to be quite simple.

Datasets for training and evaluating Complex Word Identification (CWI) systems were created and released in the SemEval 2016 competition (Paetzold and Specia, 2016) but we decided not to use them for several reasons. Although this was a CWI task, surprisingly only 4.7% of the words in the test data were identified as complex, and all the other words were viewed as simple. As a consequence, none of the systems that participated in the SemEval task managed to beat the accuracy of the “All Simple” baseline which labeled all words in the test set as simple (0.953). As noted by Paetzold and Specia (2016), the inverse problem is present in the corpus developed by Shardlow (2013b), where the “All Complex” baseline achieved higher accuracy, recall and F-scores than all other tested systems, suggesting that marking all words in a sentence as complex is the most effective approach for CWI.

Another problem in the SemEval-2016 dataset is that although the number of complex words is much higher in the training data (32%), 18% of all words were annotated as complex by only one out of 20 annotators and considered as complex. In addition to the highly different number of complex words in the training and test data, the two datasets are also imbalanced in terms of size, with only 2,237 training instances and 88,211 testing instances. These factors make this dataset a dubious choice for system training and evaluation. Comparison to the participating systems is also extremely difficult, since the best systems are ones that label most of the data as simple.

3.2.2. Complex Word Identification Experiments and Results

We compare the performance of three SVM classifiers trained with different feature sets; one classifier trained with only word-based features (**SVM Word**), one trained with both word- and context-based features (**SVM Word+Context**); and a third was trained with word, context, POS, and embedding features (**SVM-ALL**). We compare our models to two simple baselines: **Word Length**, where we simply threshold for word length in characters, considering longer words as complex; the length threshold with the best performance was 7; and *n*-**gram Frequency**: thresholding for word frequency using Google *n*-gram counts, considering more frequent words as simple; the frequency threshold with the best performance was 19,950,000. In addition, we compare our approaches to two fine-tuned models:

Model	Precision	Recall	F-Score
Word Length	0.595	0.802	0.683
n-gram Frequency	0.432	**1.0**	0.603
SVM-Word	0.787	0.724	0.754
SVM-Word+Context	0.789	0.731	0.759
SVM-ALL	0.784	0.759	0.771
RoBERTa-Word	0.785	0.706	0.743
RoBERTa-Word+Context	**0.803**	0.849	**0.825**

Table 4: Cross-validation performance for three complex word identification classifiers, along with comparisons to three baselines. Scores are calculated using **unique** words in our training data.

RoBERTa Word, where we fine-tune RoBERTa using just word-level information; and **RoBERTa Word+Context**, where we fine-tune RoBERTa using both the word and the sentence from which it came.

The results of this experiment are shown in Table 4. While *n*-gram Frequency baselines have higher recall, both our models show substantial improvements in terms of precision and overall F-score. Incorporating embedding-based features further improves performance. The explicit context-based features seem to have an ambiguous impact, in that they do not improve the performance of the SVM classifier. However, when considering the RoBERTa-based models, adding the context surrounding the word significantly improves the performance. This finding validates our initial hypothesis that there are indeed some cases where a word's complexity can be significantly impacted depending on its context.

3.3. In-context Ranking and Substitution

In order to accurately replace words in texts with simpler paraphrases and ensure that the generated sentences preserve the meaning of the original, we need to take into account the surrounding context. To perform this operation, we adapt the word embedding-based lexical substitution model of Melamud et al. (2015) to the simplification task. Vector-space models have previously been shown to effectively filter PPDB paraphrases in context while preserving the meaning of the original sentences (Apidianaki, 2016; Cocos et al., 2017).

The substitution model proposed by Melamud et al. (2015) (hereafter AddCos) quantifies the fit of a substitute word s for a target word t in a context C by measuring the semantic similarity of the substitute to the target, and the similarity of the substitute to the context:

$$AddCos(s,t,C) = \frac{\cos(s,t) + \sum_{w \in C} \cos(s,w)}{|C| + 1} \tag{3.1}$$

The vectors s and t are word embeddings of the substitute and target generated by the *skip-gram with negative sampling* model, which is also known as the Word2Vec model (Mikolov et al., 2013a). The context C is the set of context embeddings generated by *skip-gram* for words appearing within a fixed-width window of the target t in a sentence. We use a context window of 1; while this seems counter-intuitive, this is the best-performing window found in an analogous lexical substitution work (Cocos et al., 2017), and we confirm this result in Section 3.3.2. We utilize the implementation of AddCos proposed by Cocos et al. (2017)[2] with 300-dimensional word and context embeddings trained over the 4 billion words in the Annotated Gigaword corpus (Napoles et al., 2012) using the gensim word2vec package (Mikolov et al., 2013a).

Given a set of substitution candidates, the model needs to select the ones that best preserve the meaning of target words in specific contexts. We only consider content words (nouns, verbs, adjectives and adverbs) as simplification targets. For a "target word-substitute" pair, we include in the model the following features which encode the strength of the semantic relationship between them:

- **PPDB 1.0 and 2.0 scores**, which represent the overall quality of paraphrases.
- **Distributional similarity scores** calculated by Ganitkevitch et al. (2013) on the Google n-grams and the Annotated Gigaword corpora.
- **Independence probability**, that is the probability that there is no semantic entailment relationship between the paraphrase pair, as calculated by Pavlick et al. (2015).

[2]Available at `https://github.com/acocos/lexsub_addcos`

- **SimplePPDB score** (Pavlick and Callison-Burch, 2016) which reflects the confidence in the simplification rule; this is used only when considering SimplePPDB paraphrases.

Again, as this work was completed prior to the rise of pre-trained language models, we additionally will compare AddCos to a BERT-based approach. Unlike with complex word identification, here we do not perform any fine-tuning. For each target word t and its corresponding sentence S, we generate RoBERTa embeddings for S, and extract the vectors corresponding to t; here, rather than take the output of the last layer, we perform mean pooling over the last four layers similar to Devlin et al. (2019). Due to RoBERTa relying on subword units, in the case that t is represented by multiple subwords (and thus multiple vectors), we simply perform mean pooling to combine the subword unit embeddings into a single 768-dimension vector, e_t. From there, for each candidate substitute word s, we replace the instance of t in sentence S with s, resulting in a new sentence S'. From there, we again generate RoBERTa for S', and extract a vector for s, e_s in the same way as before. We then compute the cosine similarity between e_t and e_s, and then rank the candidate substitutes based on their cosine similarity.

3.3.1. Lexical Simplification Data

In our experiments, candidate substitutes for a target word are its synonyms in WordNet (Miller, 1995), and its paraphrases in the Paraphrase database (PPDB) (Ganitkevitch et al., 2013) and in SimplePPDB (Pavlick and Callison-Burch, 2016). WordNet is a network that contains manually identified semantic relationships between words, which has been widely used in substitution tasks (McCarthy and Navigli, 2007). PPDB is a collection of more than 100 million English paraphrase pairs, while SimplePPDB is a subset of PPDB which contains 4.5 million simplification rules. These corpora are discussed in more detail in Section 2.2.

To evaluate the performance of our lexical simplification model, we again create a test set from the Newsela corpus. We extract lexical simplification rules from aligned parallel

		Words with ≥1 paraphrase			All words		
Model	**Coverage**	**Top 1**	**Top 5**	**Oracle**	**Top 1**	**Top 5**	**Oracle**
WordNet frequency	0.911	0.141	0.267	0.291	0.129	0.244	0.265
SimplePPDB Score	0.935	0.180	0.403	0.669	0.168	0.377	0.626
AddCos-PPDB	**0.975**	0.196	0.444	**0.962**	0.191	0.433	**0.938**
AddCos-SimplePPDB	0.819	**0.353**	**0.601**	0.643	**0.289**	**0.492**	0.527
RoBERTa-SimplePPDB	0.819	0.161	0.487	0.643	0.132	0.397	0.527

Table 5: Performance of the lexical simplification models on the Newsela aligned test set. We present the **Coverage** of each lexico-semantic resource, the performance of each model on words with at least one paraphrase in the dataset, and the performance of each model on all words.

sentences (Xu et al., 2015) using two methods. First, we find sentence pairs with a single lexical replacement and use these word pairs as simplification instances. Next, we use a monolingual word alignment software (Sultan et al., 2014) to extract all aligned word pairs. We only consider word pairs corresponding to different lemmas (i.e. words with different base forms). Through this process, we collect a test set of 14,436 word pairs.

- **WordNet Frequency**: We extract all WordNet synonyms for a target word t, and rank them in decreasing order of Google n-gram frequency, i.e. the most frequent synonym will be ranked first and the least frequent one will be ranked last.

- **SimplePPDB Score**: We extract all SimplePPDB unigram paraphrases for t and rank them in decreasing order of their SimplePPDB score.

- **AddCos-PPDB**: We extract all PPDB synonyms for t and rank them using the AddCos model described above.

- **RoBERTa-SimplePPDB**: We extract all SimplePPDB unigram paraphrases for t, and then rank the synonyms based on their contextual RoBERTa embedding similarity to t.

3.3.2. Experiments and Results

We evaluate the performance of the lexical substitution model on our substitution evaluation dataset using Simple PPDB paraphrases. We retrieve the top suggestions made by our word

Context Window	**Top 1**	**Top 5**
0	0.180	0.403
1	**0.353**	**0.601**
2	**0.352**	0.596
3	0.334	0.590
4	0.312	0.585
5	0.291	0.581
6	0.269	0.578
7	0.264	0.577
8	0.252	0.576
9	0.247	0.574
10	0.242	0.572

Table 6: Quality of substitutions proposed by AddCos-SimplePPDB with different context window sizes as measured using Top 1 and Top 5 accuracy on the Newsela aligned test set.

embedding-based substitution model for a complex word in a sentence using the paraphrases of the word in SimplePPDB as our substitute pool (AddCos-SimplePPDB). We compare our method to three baselines:

We report the results of our experiment in Table 5. For each model, we calculate Top 1 and Top 5 accuracy scores, which show how often the gold-standard simple word was proposed as the best fitting or among the 5 highest-ranked paraphrases. In addition, we calculate the upper bound performance for each dataset (PPDB, SimplePPDB and WordNet), i.e. how often the gold-standard simple word was found as a paraphrase of the target word in the dataset. This is useful in telling us how well we could potentially do, if we could perfectly rank the paraphrases. As expected, AddCos-SimplePPDB outperforms all previous baselines, even despite the lower oracle coverage when compared with using PPDB. Somewhat surprisingly, leveraging RoBERTa performed significantly worse than AddCos; this may be because the context around a word significantly impacts a word's RoBERTa embedding, and because all substitutes were put into the same context, it made it difficult to determine which was more appropriate.

When performing this experiment, we also evaluated the impact of the context window size on the quality of the proposed substitutions. We varied the context window used by the *AddCos-SimplePPDB* model from 0 to 10. The results of this comparison are found in Table

Synonym Rank	**Good Substitution**	**Good Simplification**	**Both**
1	**0.396**	**0.280**	**0.227**
2	0.311	0.214	0.153
3	0.278	0.184	0.127
4	0.228	0.142	0.093
5	0.193	0.123	0.075
$\geq$ 1 good substitute	**0.622**	**0.553**	**0.435**

Table 7: Human judgments of our overall lexical simplification system. We give the proportion of substitutes the system ranked at positions 1 to 5 (i.e. from the top ranked to the fifth-ranked paraphrase in context) which was identified by a majority of workers as (a) a good substitute in context (Substitution); (b) simpler than the target word (Simplification); (c) both a good and simpler substitute (Both). We also show the proportion of complex words where at least one of the top 5 paraphrases satisfies these criteria in the last row.

6. As we can see, a context window of 1 results in the best performance, which confirms the finding from Cocos et al. (2017). We conducted additional experiments to filter the substitution candidates using SimplePPDB confidence scores, PPDB paraphrase quality scores, and AddCos context similarity scores, but these all resulted in a non-significant change in performance, and a significant decrease in coverage.

3.4. Overall Simplification System

We integrate our Complex Word Identification (CWI) classifier (*SVM-context*) and the substitution model that provided the best ranking in context (*AddCos-SimplePPDB*) into a simplification pipeline. The input is a complex text that needs to be simplified. The output consists of simplification suggestions for experts to choose from in order to create simpler versions of texts. The SVM-Context classifier is used to classify each content word that is not part of a named entity as either simple or complex. The lexical substitution model then gathers the SimplePPDB substitutes available for the complex target word and ranks them according to how well they fit the corresponding context. We only keep the top five suggestions made by the model as the final output.

To evaluate the performance of the overall simplification system, we used the 930 texts from the Newsela corpus that were not used for training the CWI classifier. Our model

Baseline	Simple	Complex
n-gram Frequency	dug, sled, chart, lakes, push, tight, harm	estimates, frequent, attributed, isolated, preferred, liability
Token Length	nursing, unknown, squares, feeling, teaching, strength	adorns, asylum, myriad, rigors, nutria, edible
RF-Context	malls, hungry, therefore, hears, heavily, rainy	engaging, secular, gridlock, torrent, sanctions, lobbying
SVM-Context	peacefully, favorite, amazing, websites, harmful, somewhat	swelled, entice, tether, chaotic, vessel, midst

Table 8: Examples of words that were incorrectly classified by the two best performing baselines and the RF-Context model, but were correctly classified by the SVM-Context model. The last row shows examples of words that were incorrectly classified by the SVM-Context model.

identified over 170,000 complex words with paraphrases in SimplePPDB. We again asked crowdsourced annotators to evaluate the suggestions made by *AddCos-SimplePPDB* for a random sample of 2,500 complex words on Amazon Mechanical Turk, in order to determine the number of good substitutions in context, the number of suggested paraphrases that are simpler than the target words, and the suggestions that are both simpler paraphrases and good in-context substitutes. Table 7 shows the quality of the paraphrases ranked by our system in positions from one to five. We can see that the paraphrases our system selects as the best have a higher likelihood of being both good substitutes in context and simpler than the target word. We also show the proportion of target words that had at least one good substitute in context, one simple substitute, and one good and simple substitute.

3.5. Error Analysis

In this section, we give examples of words for which our models give the correct output and the baselines fail to do so. In addition, we show examples on which our models perform poorly.

In Table 8, we consider examples that were incorrectly classified by each of the four best performing CWI models: the RF-Context and SVM-Context models, and the n-gram Frequency and Token Length baselines. In the first three rows, we show words that were correctly identified by SVM-Context, but incorrectly categorized by the two baselines and

Sentence	Gold-Standard	WordNet Frequency	SimplePPDB Score	AddCos-PPDB	AddCos-SimplePPDB
(7.1) Advocates **argue** that including women will help end harassment of female troops.	**say**	reason, fence, debate, contend, indicate	**say**, think, tell, talk, mean	contend, assert, acknowledge, insist, complain	**say**, claim, believe, suggest, debate
(7.2) But in April , detainees covered cameras used to **monitor** them.	**watch**	supervise, proctor, admonisher	find, meet, give, try, allow	track, manipulate, control, analyze, supervise	track, control, check, **watch**, follow
(7.3) Similarly , police can investigate cases and have the **authority** to seize animals.	**power**	agency, potency, bureau, assurance	force, control, permission, office, limit	jurisdiction, discretion, right, prerogative, ability	**power**, responsibility, body, agency

Table 9: Examples of the top-5 substitutes for our three baselines and our best model (AddCos-SimplePPDB). We also provide the gold-standard simplification (Gold-Standard).

RF-Context; in the last row, we show words incorrectly classified by SVM-Context. We observe that the n-gram Frequency model tends to incorrectly classify relatively short words that are rare in the Google n-gram corpus as complex. On the other end, the Token Length model shows that using this feature alone leads to incorrectly identifying shorter words such as “adorn” and “myriad” as simple, when these words are relatively complex.

Table 9 presents examples of substitution where the baseline systems did not find the correct paraphrase, but AddCos-SimplePPDB did. As we have mentioned, even when a model did not find the gold-standard paraphrase, they sometimes did find a different paraphrase that works well in the context. In Example 7.2, the top paraphrase proposed by both AddCos-PPDB and AddCos-Simple PPDB for the word “monitor” is “track”, which is a reasonable substitute. On the other hand, in Example 7.3, AddCos-Simple PPDB model was able to identify a good simple substitute, when none of the other models were able to identify a suitable word with comparable complexity.

Finally, Table 10 shows examples of output of the overall simplification system. Here, the **blue** word is a word that our CWI classifier identified as complex (for simplicity, we only look at one complex word per sentence). From there, we consider the five top-ranked substitutes proposed by AddCos-Simple PPDB, and show which were identified by the majority of

Sentence	Bad Substitutions
(8.1) Officials will offer the lunch program at **elementary** schools.	basic
(8.2) Russian poultry is more expensive, and U.S. producers enjoy numerous cost **advantages**.	prospect, benefits, revenue, merit, feature
(8.3) Although the calculus may be different with Syrian **refugees**, the parallel for me is politics.	life, right, return, shelter, million
(8.4) He saw them bring in animals to a university, where they'll be cared for and put up for **adoption**.	acceptance, passage, approval, endorsement

Table 10: Examples of words and their context where our model fails to provide any good replacements.

annotators as good substitutes for the target word, simpler than the target, good simpler substitutes, and bad substitutes. After reviewing the examples where our system failed to generate acceptable substitutions for the identified complex words, we identified four main categories of errors:

- The identified complex term is part of a phrase and no substitution is acceptable. For example, in Example 8.1, *Elementary*, *Middle* or *High School* is a description of the type of school. *Elementary School* has an alternative name in some cases but *High School* should never become *Tall School*.

- The complex word has no simpler synonym that would be a good substitute. The difficulty of the word might reside in its meaning which can be unknown to the reader. In Example 8.3, it would be more useful to point to the definition of *refugees*.

- The complex word is part of a predicate with arguments that are not accessible to our model. In Example 8.4, the intended meaning of *adoption*, human adoption, is hard to capture in the vicinity of the complex word.

- Finally, in some cases, our annotators were quite strict in admitting a substitute. In Example 8.2, for example, *cost merit* would not be syntactically correct but *cost merits* would be acceptable.

3.6. Summary

In this chapter, we presented a model for simplification that first identifies complex words in texts, and then ranks lexical simplification candidates according to their adequacy in these specific contexts. We performed experiments showing that our model makes correct simplification suggestions 35% of the time as measured by top-1 accuracy (versus 20% of the time for the best baseline), and produces a good substitution in its top-5 predictions 60% of the time (versus 44% for the best baseline). We performed a detailed error analysis that suggests future improvements, e.g. not replacing words within collocations like *elementary school*, and extending the context model to include the arguments of words that are going to be simplified.

CHAPTER 4 : Modeling and Evaluation of Sentence Simplification

4.1. Introduction

In the previous chapter, we focused on complex words, and their substitution with simpler words that preserve the original in-context meaning. In this chapter, we instead consider the task of simplifying an entire sentence, a task known as sentence simplification. Most recent sentence simplification research has approached this task as a monolingual machine translation problem, where the goal is to transform a complex English sentence into a simpler sentence that preserves the meaning of the original sentence (Zhu et al., 2010; Narayan and Gardent, 2014).

This chapter includes two sections. In the first section, we consider applying sequence-to-sequence (Seq2Seq) models to the task of sentence simplification, and propose various modifications to extend the generic framework during training, inference, and post-inference. We also provide an extensive error analysis to show where current sentence simplification models fall short. In the second section, we discuss how to fine-tune a BERT-based model on fluency, adequacy, and complexity simultaneously in order to predict the overall quality of sentence simplification system output without the need for references.

4.2. Complexity-Weighted Loss and Diverse Re-Ranking for Sentence Simplification

If we frame simplification as a monolingual translation problem, a natural approach is to follow a machine translation approach and apply sequence-to-sequence (Seq2Seq) models (Sutskever et al., 2014; Luong et al., 2015; Vaswani et al., 2017). Seq2Seq models learn mappings from one sequence to another using a neural network, and have shown state-of-the-art performance on other related monolingual natural language processing generation tasks, including text summarization (Nallapati et al., 2016) and dialog systems (Vinyals and Le, 2015).

One of the main limitations in applying standard Seq2Seq models to simplification is that these models tend to copy directly from the original complex sentence too often, as this is the most common operation in simplification. Several recent efforts have attempted to alleviate this problem using reinforcement learning (Zhang and Lapata, 2017) and memory augmentation (Zhao et al., 2018), but these systems often still produce outputs that are longer than the reference sentences. To avoid this problem, we propose to extend the generic Seq2Seq framework at both training and inference time by encouraging the model to choose simpler content words, and by effectively choosing an output based on a large set of candidate simplifications. The main extensions from this work can be summarized as follows:

- We propose a custom loss function to replace standard cross entropy probabilities during training, which takes into account the complexity of content words.
- We include a similarity penalty at inference time to generate more diverse simplifications, and we further cluster similar sentences together to remove highly similar candidates.
- We develop methods to re-rank candidate simplifications to promote fluency, adequacy, and simplicity, helping the model choose the best option from a diverse set of sentences.

An analysis of each individual components reveals that of the three contributions, re-ranking simplifications at post-decoding stage brings about the largest benefit for the simplification system. We compare our model to several state-of-the-art systems in both an automatic and human evaluation settings, and show that the generated simple sentences are shorter and simpler, while remaining competitive with respect to fluency and adequacy. We also include a detailed error analysis to explain where the model currently falls short and provide suggestions for addressing these issues.

4.2.1. Seq2Seq Approach

Complexity-Weighted Loss Function

Standard Seq2Seq models use cross entropy as the loss function at training time. This only takes into account how similar our generated tokens are to those in the reference simple sentence, and not the complexity of said tokens. Therefore, we first develop a model to predict word complexities, and incorporate these into a custom loss function.

Extending the binary complex word identification model from Chapter 3, we train a linear regression model using length, number of syllables, and word frequency; we also include Word2Vec embeddings (Mikolov et al., 2013b). We leverage the Newsela corpus to collect data for this task by extracting word counts in each of the five reading levels, labeling each word with a complexity level based on those counts. We propose using Algorithm 3 to obtain the complexity label for each word w, where l_w represents the level given to the word, and c_{w_i} represents the number of times that word occurs in level i.

Algorithm 3 Word Complexity Data Collection

```
1: procedure DATA COLLECTION
2:     $l_w \leftarrow 4$
3:     for $i \in \{3, 0\}$ do
4:         if $c_{w_i} \geq 0.7 * c_{w_{i+1}}$ then
5:             if $c_{w_i} \geq 0.4 * c_{w_4}$ then
6:                 $l_w \leftarrow i$
       return $l_w$
```

Here, we initially label the word with the most complex level, 4. If at least 70% of the instances of this word is preserved in level 3, we reassign the label as level 3; if the label was changed, we then do this again for progressively simpler levels. As examples, Algorithm 3 labels "pray", "sign", and "ends" with complexity level 0, and "proliferation", "consensus", and "emboldened" with complexity level 4. We split the data extracted from Algorithm 3 into Train, Validation and Test sets (90%, 5% and 5%, respectively, and use them for training and evaluating the complexity prediction model. Note that we also tried continuous

Model	**Correlation**	**Mean Squared Error**					
		Overall	**0**	**1**	**2**	**3**	**4**
Frequency	-0.031	1.9	2.24	0.38	0.64	3.01	6.99
Length	0.344	1.51	1.03	**0.32**	0.89	2.95	6.90
LinReg	**0.659**	**0.92**	**0.92**	0.39	**0.49**	**1.17**	**3.27**

Table 11: Pearson Correlation, Overall Mean Squared Error (MSE), and MSE by complexity level for our word-level complexity prediction model. We compare to length-based and frequency-based baselines.

rather than discrete labels for words by averaging frequencies, but found that this increased the noise in the data. For example, "the" and "dog" were incorrectly labeled as level 2 instead of 0, since these words are seen frequently across all levels.

We report the Mean Squared Error (MSE) and Pearson correlation on our test set in Table 11. We compare our model (LinReg) to the two strongest baselines from Chapter 3, which predict complexity using log Google n-grams frequency (Brants and Franz, 2006) and word length, respectively.

Once we have a reliable way of making word-level complexity predictions, we then propose a method that modifies cross entropy loss to up-weight simple words while down-weighting more complex words. More formally, the probabilities of our simplified loss function can be generated by the process described in Algorithm 4. Since our word complexities are originally from 0 to 4, with 4 being the most complex, we need to reverse this ordering and add one, so that more complex words and non-content words are not given zero probability. In this algorithm, we denote the original probability vector as $\mathbf{CE}$, our vocabulary as $\mathbf{V}$, the predicted word complexity of a word v as $score_v$, the resulting weight for a word as w_v, and our resulting weights as $\mathbf{SCE}$, which we then normalize and convert back to logits.

Here, α is a parameter we can tune during experimentation. Note that we only upweight simple content words, not stopwords or entities.

Algorithm 4 Simplified Loss Function

```
1: procedure Simplified Loss
2:     CE ← softmax($logits_{CE}$)
3:     for $v \in \mathbf{V}$ do
4:         $score_v \leftarrow WordComplexity(v)$
5:         if $v$ is a content word then
6:             $w_v \leftarrow (4 - s_v) + 1$
7:         else
8:             $w_v \leftarrow 1$
9:     $w_v \leftarrow \left(\frac{w_v}{\sum_{v \in V} w_v}\right)^{\alpha}$ for $v \in \mathbf{V}$
10:    SCE ← CE · w
       return SCE
```

Diverse Candidate Simplifications

To increase the diversity of our candidate simplifications, we apply a beam search scoring modification proposed in Li et al. (2016b). In standard beam search with a beam width of b, given the b hypotheses at time $t-1$, the next set of hypotheses is generated by first selecting the top b candidate expansions from each hypothesis. These $b \times b$ hypotheses are then ranked by the joint probabilities of their sequence of output tokens, and the top b according to this ranking are chosen.

We observe that candidate expansions from a single parent hypothesis tend to dominate the search space over time, even with a large beam. To increase diversity, we apply a penalty term based on the rank of a generated token among the b candidate tokens from its parent hypothesis.

If Y_{t-1}^{j} is the j^{th} top hypothesis at time $t-1$, $j \in [1..b]$, and $y_t^{j,j'}$ is a candidate token generated from Y_{t-1}^{j}, where $j' \in [1..b]$ represents the rank of this particular token among its siblings, then our modified scoring function is as follows (here, δ is a parameter we can tune during experimentation):

$$S(Y_{t-1}^{j}, y_t^{j,j'}) = \log p(y_1^{j}, \ldots, y_{t-1}^{j}, y_t^{j,j'} | x) - j' * \delta \tag{4.1}$$

Extending the work of Li et al. (2016b), to further increase the distance between candidate simplifications, we can cluster similar sentences after decoding. To do this, we convert each candidate into a document embedding using Paragraph Vector (Le and Mikolov, 2014), cluster the vector representations using k-means, and select the sentence nearest to the centroids. This allows us to group similar sentences together, and only consider candidates that are relatively more different. Note that this is only one of many possible diverse decoding approaches, I explore these in more detail in Chapter 5.

Re-ranking Diverse Candidates

Generating diverse sentences is helpful only if we are able to effectively re-rank them in a way that promotes simpler sentences while preserving fluency and adequacy. To do this, we propose three ranking metrics for each sentence i:

- **Fluency** (f_i): We calculate the perplexity based on a 5-gram language model trained on English Gigaword v.5 (Parker et al., 2011) using KenLM (Heafield, 2011).
- **Adequacy** (a_i): We generate Paragraph Vector representations Le and Mikolov (2014) for the input sentence and each candidate and calculate the cosine similarity.
- **Simplicity** (s_i): We develop a sentence complexity prediction model to predict the overall complexity of each sentence we generate.

To calculate sentence complexity, we modify a Convolutional Neural Network (CNN) for sentence classification (Kim, 2014) to make continuous predictions. We use aligned sentences from the Newsela corpus (Xu et al., 2015) as training data, labeling each with the complexity level from which it came. We normalize each individual score between 0 and 1, and calculate a final score as follows:

$$score_i = \beta_f f_i + \beta_a a_i + \beta_s s_i \tag{4.2}$$

Parameter	**Value**	**Parameter**	**Value**
Batch size	86	Embeddings	300:300
RNN hidden units	256	LR	0.001
RNN attention	dot	LR reduce	0.7
# of layers	2	Loss	CE
RNN type	LSTM	Min Epochs	1
Dropout inputs	0.2	Max epochs	30
Dropout states	0.2	Max updates	500000
Min vocab freq	3	# Last params	5
Max length	85	Optimizer	Adam
Label smoothing	0	Seed	13

Table 12: Training Hyperparameters for the baseline Seq2Seq model and our extended model.

We tune these weights (β) on our validation data during experimentation to find the most appropriate combinations of re-ranking metrics.

4.2.2. Experiments

We train our models on the Newsela Corpus (Xu et al., 2015). Following Zhang and Lapata (2017), we exclude sentence pairs corresponding to levels 4-3, 3-2, 2-1, and 1-0, where the simple and complex sentences are just one level apart, as these are too close in complexity. After this filtering, we are left with 94,208 training, 1,129 validation, and 1,077 test sentence pairs; these splits are the same as Zhang and Lapata (2017). We preprocess our data by tokenizing and replacing named entities using CoreNLP (Manning et al., 2014).

For our experiments, we use Sockeye, an open source Seq2Seq framework built on Apache MXNet (Hieber et al., 2017; Chen et al., 2015). In this model, we use LSTMs with attention for both our encoder and decoder models. We attempt to match the hyperparameters described in Zhang and Lapata (2017) as closely as possible; as such, we use 300-dimensional pretrained GloVe word embeddings (Pennington et al., 2014), and Adam optimizer (Kingma and Ba, 2015). Table 12 shows a comprehensive list of all hyperparameters we used when training our default Seq2Seq model. This list includes learning rate (LR), learning rate reduction rate (LR reduce), size of embeddings (Embeddings), loss function (loss, we use CE to represent Cross Entropy), among others.

For our extensions to the standard Seq2Seq framework, we use nearly all the same parameters, the only exception being that we change the loss function from cross entropy to our custom complexity-weighted loss function. With this loss, we use $\alpha = 2$ during training. At inference time, we set the beam size $b = 100$, and the similarity penalty $d = 1.0$. After inference, we set the number of clusters to 20, and we compare two separate re-ranking weightings: one which uses fluency, adequacy, and simplicity (FAS), where $\beta_f = \beta_a = \beta_s = \frac{1}{3}$; and one which only uses fluency and adequacy (FA), where $\beta_f = \beta_a = \frac{1}{2}$ and $\beta_s = 0$. Note that our best model uses FA weights.

We compare our models to the following baselines:

- **Hybrid** performs sentence splitting and deletion before simplifying with a phrase-based machine translation system (Narayan and Gardent, 2014).
- **DRESS** is a Seq2Seq model trained with reinforcement learning which integrates lexical simplifications (Zhang and Lapata, 2017).[1]
- **DMASS** is a Seq2Seq model which integrates the transformer architecture and additional simplifying paraphrase rules (Zhao et al., 2018).[2]

We also present results on several variations of our models, to isolate the effect of each individual improvement. **Seq2Seq** is a standard sequence-to-sequence model with attention and greedy search. **Seq2Seq-Loss** is trained using our complexity-weighted loss function and greedy search. **Seq2Seq-FA** uses beam search, where we re-rank all sentences using fluency and adequacy (FA weights). **Seq2Seq-Cluster-FA** clusters the sentences before re-ranking using FA weights. **Seq2Seq-Diverse-FA** uses diversified beam search, re-ranking using FA weights. **Seq2Seq-All-FAS** uses all contributions, re-ranking using fluency, adequacy, and simplicity (FAS weights). Finally, **Seq2Seq-All-FA** integrates all modifications we propose, and re-ranks using FA weights.

[1]For Hybrid and DRESS, we use the generated outputs provided in Zhang and Lapata (2017). We made a significant effort to rerun the code for DRESS, but were unable to do so.

[2]For DMASS, we ran the authors' code on our data splits from Newsela, in collaboration with the first author to ensure an accurate comparison.

Model	**SARI**	Oracle	**Len**	**FKGL**	**TER**	**Ins**	**Edit**
Complex	–	–	23.1	11.14	0	0	–
Hybrid	33.27	–	12.4	7.82	0.49	0.01	–
DRESS	36.00	–	14.4	7.60	0.44	0.07	–
DMASS	34.35	–	15.1	7.40	0.59	0.28	–
Seq2Seq	36.32	–	16.1	7.91	0.41	0.23	–
Seq2Seq-Loss	36.03	–	16.4	8.11	0.40	**0.31**	–
Seq2Seq-FA	36.47	54.01	7.6	6.42	0.73	0.01	7.28
Seq2Seq-Cluster-FA	**37.22**	50.36	9.1	6.49	0.68	0.05	7.55
Seq2Seq-Diverse-FA	35.36	52.65	7.5	5.97	**0.78**	0.07	**8.22**
Seq2Seq-All-FAS	36.30	50.40	9.1	**5.37**	0.68	0.05	7.56
Seq2Seq-All-FA	37.11	50.40	10.8	6.42	0.61	0.07	7.56
Reference	100	–	12.8	6.90	0.67	0.42	–

Table 13: Comparison of our models to baselines and state-of-the-art models using SARI. We also include oracle SARI scores (Oracle), given a perfect re-ranker. Seq2Seq-All-FA is significantly better than the baselines using a student t-test ($p < 0.05$). We also calculate average sentence length, FKGL, TER score compared to input, number of insertions, and average edit distance (Edit) between candidate sentences for applicable models.

Following previous work (Zhang and Lapata, 2017; Zhao et al., 2018), we use SARI as our main automatic metric for evaluation (Xu et al., 2016).[3] SARI calculates how often a generated sentence correctly keeps, inserts, and deletes n-grams from the complex sentence, using the reference simple sentence as the gold-standard, where $1 \leq n \leq 4$. We also calculate oracle SARI, where appropriate, to show the score we could achieve if we had a perfect re-ranking model. Our results are reported on the left side of Table 13.

Our best models outperform previous state-of-the-art systems, as measured by SARI. When used separately, re-ranking and clustering results in the largest improvements on this metric. Our loss and diverse beam search methods have more ambiguous effects, especially when combined with the former two; note however that including diversity before clustering does slightly improve the oracle SARI score.

We calculate several descriptive statistics on the generated sentences and report the results on the right side of Table 13. We observe that our models produce sentences that are much shorter and lower reading level, according to Flesch-Kincaid grade level (FKGL) (Kincaid et al., 1975), while making more changes to the original sentence, according to Translation

[3]To compute SARI, we use the original script provided by (Xu et al., 2016).

Model	Fluency	Adequacy	Simplicity	All
Hybrid	2.79*	2.76	2.88*	2.81*
DRESS	**3.50**	**3.11***	3.03	**3.21***
DMASS	2.59*	2.15*	2.50*	2.41*
Seq2Seq-All-FAS	3.35	2.50*	**3.11**	2.99
Seq2Seq-All-FA	3.38	2.66	**3.08**	3.04
Reference	3.82*	3.23*	3.29*	3.45*

Table 14: Average ratings of crowdsourced human judgments on fluency, adequacy and complexity. Ratings significantly different from Seq2Seq-All-FA are marked with * ($p < 0.05$); statistical significance tests were calculated using a student t-test. We provide 95% confidence intervals for each rating in the appendix.

Error Rate (TER) (Snover et al., 2006). In addition, we see that the customized loss function increases the number of insertions made, while both the diversified beam search and clustering techniques individually increase the distance between sentence candidates.

While SARI has been shown to correlate with human judgments on simplicity, it only weakly correlates with judgments on fluency and adequacy (Xu et al., 2016). Furthermore, SARI only considers simplifications at the word level, while we believe that a simplification metric should also take into account sentence structure complexity.

Due to limitations of current automatic metrics, we also choose to elicit human judgments on 200 randomly selected sentences to determine the relative overall quality of our simplifications. For our first evaluation, we ask native English speakers on Amazon Mechanical Turk to evaluate the fluency, adequacy, and simplicity of sentences generated by our systems and the baselines, similar to Zhang and Lapata (2017). Each annotator rated these aspects on a 5-point Likert Scale. These results are found in Table 14.

As we can see, our best models substantially outperform the Hybrid and DMASS systems. Note that DMASS performs the worst, potentially because the transformer model is a more complex model that requires more training data to work properly. Comparing to DRESS, our models generate simpler sentences, but DRESS better preserves the meaning of the original sentence.

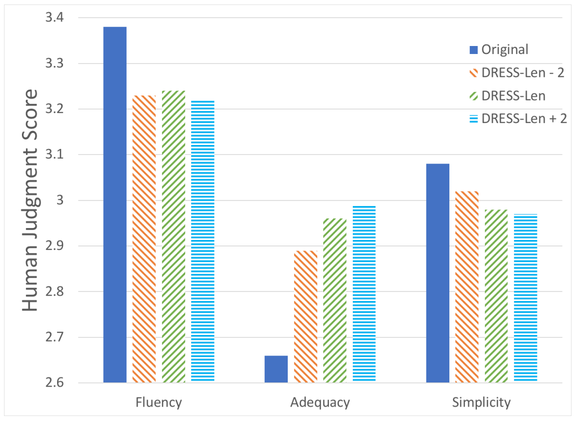


Figure 7: Effect of length on human judgments.

To further investigate why this is the case, we know from Table **??** that sentences generated by our model are overall shorter than other models, which also corresponds to higher TER scores. Napoles et al. (2011) notes that on sentence compression, longer sentences are perceived by human annotators to preserve more meaning than shorter sentences, controlling for quality. Thus, the drop in human-judged adequacy may be related to our sentences' relatively short lengths.

To test that this observation also holds true for simplicity, we took the candidates generated by our best model, and after re-ranking them as before, we selected three sets of sentences:

- **MATCH-Dress0**: Highest ranked sentence with length closest to that of DRESS (DRESS-Len); average length is 14.10.
- **MATCH-Dress+2**: Highest ranked sentence with length closest to (DRESS-Len + 2);
  average length is 15.32.
- **MATCH-Dress-2**: Highest ranked sentence with length closest to (DRESS-Len - 2); average length is 12.61.

The average fluency, adequacy, and simplicity from human judgments on these new sentences are shown in Figure 7, along with those ranked highest by our best model (Original). As expected, meaning preservation does substantially increase as we increase the average sentence length, while simplicity decreases. Interestingly, fluency also decreases as sentence length increases; this is likely due to our higher-ranked sentences having greater fluency, as defined by language model perplexity.

Error Analysis

To gain insight in what aspects of the simplification process are challenging to our model, we present the most recurring types of errors from our test set.

1. Long and complex sentences with multiple clauses

    (a) *Complex*: Turkey has long enshrined the secular ideals of founding father Mustafa Kemal Ataturk, particularly in an education system that until recently banned Islamic headscarves in schools and made schoolchildren begin the day reciting an oath of allegiance to Ataturk's legacy.
    Reference: Schools in Turkey had banned headscarves.
    Simple: They made schoolchildren to Ataturk's history.

    (b) *Complex*: And Wal-Mart, which imports more fruits and vegetables from Mexico than any other U.S. company, announced its effort to force improvements up and down its supply chain.
    Reference: Experts said Wal-Mart is an important company.
    Simple: Wal-Mart used more fruits and vegetables from the company.

2. Need for anaphora resolution

    (a) *Complex*: He is the creative director of Rethink Leisure & Entertainment , which is working on several projects in China and elsewhere in Asia .
    Reference: He is with Rethink Leisure & Entertainment.
    Simple: He is working on several projects in China.

    (b) *Complex*: Teachers there say Richie reads like a high school student.

Reference: He reads like a high school student.

Simple: Richie says he is a high school student.

3. Simplifying the wrong part of the sentence

   (a) *Complex*: Parks deliberately maintained her image as shy and proper, said Adrienne Cannon, an expert on African-American history.

   Reference: Adrienne Cannon studies African-American history.

   Simple: She is an expert on African-American history.

   (b) *Complex*: His father owned the home when the lava flowed slowly to the coast.

   Reference: His father still owned the home.

   Simple: The river cut slowly to the coast.

4. Poor substitution due to word embeddings proximity

   (a) *Complex*: Just until recently, the education system had banned Islamic **headscarves** in schools and made schoolchildren begin the day reciting a pledge of allegiance to Ataturk's legacy.

   Reference: Schools in Turkey had banned headscarves.

   Simple: Just until recently, the education officials had banned Islamic **flags**.

   (b) *Complex*: In Beijing kite circles, Fei is widely known as the elder **statesman**.

   Reference: In Beijing, Fei is widely known as an expert.

   Simple: In Beijing, Fei is considered a **doctor**.

5. Training confusion due to misalignments in training corpus

   (a) *Complex*: He heard the applause.

   Reference: The winning word was "magician."

   Simple: But he heard the song.

   (b) *Complex*: More than 300,000 students across the country signed up to participate.

   Reference: Rogers Middle School took part in the program.

   Simple: There were more than 300,000 students.

6. Acceptable simplifications that were not available in the training

   (a) *Complex*: He later tried to buy his freedom.
   Reference: Henson tried to buy his freedom later.
   Simple: The man tried to buy his freedom.

   (b) *Complex*: Middle school was a rough couple of years for Talmus' daughter, Lili Smith.
   Reference: She had a hard time in middle school.
   Simple: School was a rough couple of years.

Attempting to rewrite very long and complex sentences resulted to consistent errors, as shown in 1a and 1b. This observation in combination with the examples of mis-alignments in the training corpus (5a and 5b) indicate that we either need to improve the alignments such the model can capture that the simplification process involves in many cases splitting a sentence and then simplifying or train to learn when to split first and then attempt rewriting.

The next two types of errors show failure in capturing discourse level meaning: a) errors due to failed pronoun resolution, shown in 2a and 2b, and b) errors due to the most important part of the sentence being left out, shown in 3b and 3b. In these cases, the sentences were not bad, but the information was assigned to the wrong referent, or important meaning was left out. In 4a and 4b, the substitution is clearly semantically related to the target, but changes the meaning. Finally, there were examples of acceptable simplifications, as in 6a and 6b, that were classified as errors because they were not in the gold data. We provide additional examples for each error category in the appendix.

To improve the performance of future models, we see several options. We can improve the original alignments within the Newsela corpus, particularly in the case where sentences are split. Prior to simplification, we can use additional context around the sentences to perform anaphora resolution; at this point, we can also learn when to perform sentence splitting; this has been done in the Hybrid model (Narayan and Gardent, 2014), but has not yet been incorporated into neural models. Finally, we can use syntactic information to ensure the main clause of a sentence is not removed.

4.3. Simple-QE: Better Automatic Quality Estimation for Text Simplification

In many text generation tasks, it is generally necessary to evaluate model quality using human judgments. However, this is expensive and time consuming, and is untenable when testing many model variations. Thus, many works have attempted to develop metrics to automatically evaluate the quality of generation models. Previous evaluation metrics (Papineni et al., 2002; Xu et al., 2016) estimate the quality of generated texts by comparing them to human-written simplifications, which restricts their use to settings where such references are available. In addition, comparing simplifications to a single reference is often too restrictive, as most texts can be simplified in a variety of ways.

We propose a model for measuring the quality of automatically generated simplifications, which does not require human references. Our model, Simple-Quality Estimation (Simple-QE), adapts the BERT-based summary QE model Sum-QE (Xenouleas et al., 2019), to the simplification setting. As opposed to summaries – which contain specific pieces of information from the original text and omit unimportant passages – simplified text typically expresses all the content present in the original text using simpler words and structures. Both types of text, however, need to fulfill some linguistic quality constraints in order to be useful, such as being grammatical and well-formed. We show that Simple-QE correlates well with human judgments of linguistic quality on system output produced by simplification systems. In addition, we adapt our model to make reasonable complexity predictions at both the sentence and document level. Our models can be used to optimize both simplification system development and the process of writing manual simplifications.

4.3.1. Methodology

To estimate the overall quality of a simplification system output, we focus on three linguistic aspects:

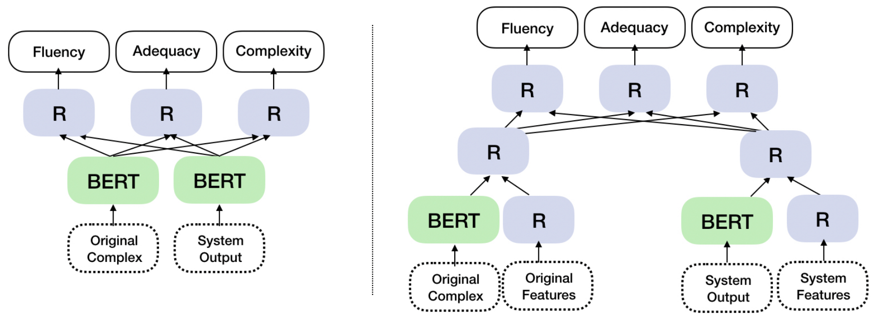


Figure 8: The Simple-QE architecture, with and without adding numeric features corresponding to the original complex sentence (Original Features) and the system output (System Features). *R* denotes a regressor layer.

- **Fluency**: How well-formed the system out is.
- **Adequacy**: How well the system output preserves the meaning of the original text.
- **Complexity**: How much simpler the system out is than the original text. sentence.

We adapt the architecture proposed by Xenouleas et al. (2019) in their Sum-QE model, which extends the BERT fine-tuning process (Devlin et al., 2019) to rate summaries with respect to five linguistic qualities. We expect Fluency, in our setting, to align well with Grammaticality as addressed by Sum-QE. In the case of Adequacy and Complexity, since judgments are relative (e.g. is the generated text *simpler than* the original text? does it convey *the same* meaning?), we need to also consider the original complex text.

Xenouleas et al. (2019) use BERT as the main encoder and fine-tune it in three ways, one single-task and two multi-task approaches:

- **Single Task (S-1)**: Train k models on each annotation type, where k is the number of linguistic qualities; here, $k = 3$.
- **Multi Task-1 (M-1)**: Train one model with a single regressor to predict k annotations.
- **Multi Task-k (M-3)**: Train one model with k separate regressors, each correspond-

ing to an individual annotation.

To adapt Sum-QE to simplification, we extend the architecture to take into account the original complex sentence. We do so by passing the original complex sentence and simplification system output through the BERT architecture separately. We concatenate the resulting embedding representations, and pass them through a final dense linear regressor layer R to predict each linguistic quality score. Our adaptation of the Sum-QE Multi Task-3 (M-3) approach is described on the left side of Figure 8.

To further adapt the QE model to our task, we also attempt to incorporate task-specific features: the average **Length** of content words in a sentence in characters and in **Syllables**, their **Unigram Frequency**,[4] the **Sentence Length**, and the syntactic **Parse Height**. We pass these features extracted from the original complex sentence separately through a linear layer, before concatenating them with the BERT embeddings of the sentence. We do the same for the system output. The right side of Figure 8 describes this architecture.

4.3.2. QE Experiments on System Output

Our test data consists of human judgments collected in Section **??** on generated simplifications for 200 Newsela sentences (Xu et al., 2015). For each sentence, outputs from six simplification models were considered: vanilla Sequence-to-Sequence (Seq2Seq) (Nisioi et al., 2017), Seq2Seq with reinforcement learning (Zhang and Lapata, 2017), memory-augmented Transformer (Zhao et al., 2018), and three variations of our Seq2Seq model with post-training re-ranking from Section 4.2. Annotators were asked to rate the Fluency, Adequacy, and Complexity of each system output on a 5-point Likert Scale. Note that we do not use the QE dataset introduced by Stajner et al. (2016), as it focuses on small-scale lexical changes – similar to the Turk dataset (Xu et al., 2016) – while current neural models adopt a more holistic approach.

We compare the Simple-QE model to baselines that use the simplification-specific features

[4]We use the average log unigram frequency from the Google n-gram corpus (Brants and Franz, 2006).

Model	Fluency			Adequacy			Complexity		
	ρ	τ	r	ρ	τ	r	ρ	τ	r
BLEU	0.167	0.116	0.183	0.278	0.195	0.305	0.046	0.033	0.057
SARI	0.140	0.098	0.133	0.192	0.133	0.207	0.002	0.002	0.013
BERT LM	0.443	0.314	0.411	0.397	0.277	0.350	0.277	0.195	0.249
BERT Sim	0.288	0.202	0.278	0.530	0.382	0.533	0.095	0.066	0.085
Sum-QE S-1	0.603	0.430	0.619	0.484	0.346	0.489	0.392	0.275	0.388
Sum-QE M-1	0.626	0.446	0.632	0.535	0.384	0.541	0.403	0.282	0.407
Sum-QE M-3	0.627	0.448	0.638	0.519	0.371	0.518	0.421	0.297	0.418
Simple-QE S-1	0.619	0.443	0.635	0.545	0.393	0.549	0.438	0.309	0.436
Simple-QE M-1	0.628	0.448	0.643	**0.623**	**0.454**	**0.630**	0.435	0.305	0.433
Simple-QE M-3	**0.638**	**0.459**	**0.648**	0.604	0.439	0.612	**0.459**	**0.325**	**0.464**

Table 15: Correlations with human judgements on Fluency, Adequacy and Complexity of simplification system output; Spearman's ρ, Kendall's τ and Pearson's r are reported.

described in Section 4.3.1, quality estimates provided by BLEU (Papineni et al., 2002) and SARI (Xu et al., 2016), and three additional BERT-based baselines.

- **BERT as Language Model (BERT LM)**: Given a sentence, we mask each token and predict the likelihood of the true word occurring in this context; this captures Fluency.[5]

- **BERT embedding similarity (BERT Sim)**: We convert the original and simplified texts into sentence-level BERT vector representations via mean pooling, and compute their cosine similarity; this estimates Adequacy.

- **Sum-QE**: We apply Sum-QE directly, fine-tuning only on annotated system output.

For Sum-QE and Simple-QE, we perform 10-fold cross validation, combining the results to compute the overall correlation.

The results are shown in Table 15. Simple-QE correlates better with human judgments than the baseline models tested. The correlation of BLEU and SARI with human judgments is particularly low, especially for Complexity. This is not surprising, given that SARI mainly addresses lexical simplification, while recent models approach simplification more holistically.

[5]https://github.com/xu-song/bert-as-language-model

The three versions of Simple-QE perform similar to Sum-QE on Fluency, where the model does not need to access the original complex sentence. The difference between the two models is more noticeable for Adequacy and Complexity, where accessing the original sentence actually helps Simple-QE make more reasonable estimates. From the three versions of Simple-QE tested, the multi-task versions perform better than the single task on all three qualities tested. The **BERT LM** and **BERT Sim** baselines perform well on Fluency and Adequacy, as expected, but fall short on the other aspects of simplification. As shown in Table 15, adding numeric features do not improve performance. This may be because the most predictive features, e.g. sentence length, are already implicitly learned by BERT, as recent probing works have shown that the syntax of a sentence can be extracted from contextual word embeddings with reasonable accuracy (Hewitt and Manning, 2019).

4.4. Summary

In this chapter, we first present a novel Seq2Seq framework for sentence simplification. We contribute three major improvements over generic Seq2Seq models: a complexity-weighted loss function to encourage the model to choose simpler words; a similarity penalty during inference and clustering post-inference, to generate candidate simplifications with significant differences; and a re-ranking system to select the simplification that promotes both fluency and adequacy. Our model outperforms previous state-of-the-art systems according to SARI, the standard metric for simplification used in our evaluation, while we highlight the issues with the current automatic metrics. More importantly, while previous models generate relatively long sentences, our model is able to generate shorter and simpler sentences, while remaining competitive regarding human-evaluated fluency and adequacy. Finally, we provide a qualitative analysis of the improvements introduced by our specific contributions, discuss the effect of sentence length on human-evaluated meaning preservation, and the shortcomings of our model as insights for future research.

In the second part of this chapter, we present Simple-QE, a quality estimation model for simplification. We have shown that extending Sum-QE (Xenouleas et al., 2019) to include the reference complex sentence significantly improves predictions on Adequacy and Complexity. QE systems can be useful for evaluating the overall quality of model output without requiring expensive human annotations or references. Future simplification systems can incorporate Simple-QE into the optimization process, similar to how SARI was incorporated into a Seq2Seq network (Zhang and Lapata, 2017).

From the modifications we applied to standard Seq2Seq architectures, we found that the ones that improved performance on sentence simplification were incorporating a diverse decoding strategy, followed by re-ranking the resulting candidates. As we note, our methodology for sentence simplification is only one of many strategies that have recently been developed, as many open-ended language generation tasks can greatly benefit from multiple unique but valid candidate outputs. In the following chapter, we further pursue this idea of diversity, and perform an exhaustive comparison of current diverse decoding techniques on two tasks: conversational dialogue systems and image captioning. While these tasks have different goals than sentence simplification, all three fields share the fact that they benefit from re-ranking or combining diverse candidate outputs.

CHAPTER 5 : Comparison of Diverse Decoding Methods from Conditional Language Models

5.1. Introduction

In the previous chapter, we discussed how to more effectively apply Sequence-to-Sequence models to the task of sentence simplification, and found that encouraging diversity at inference time led to significant improvements. In this chapter, we pursue further the idea of diverse decoding, and perform an exhaustive comparison of current diverse decoding techniques.

As discussed in Section 2.5.1, conditional neural language models output a probability distribution over the next token in the output sequence, given the input and the previously predicted tokens. Since computing the overall most likely output sequence is intractable, early work in neural machine translation found that beam search is an effective strategy to heuristically sample sufficiently likely sequences from these probabilistic models (Sutskever et al., 2014). However, for more open-ended tasks, beam search is ill-suited to generating a set of diverse candidate sequences; this is because candidates output by a large-scale beam search often only differ in punctuation and minor morphological variations (Li and Jurafsky, 2016).

The term "diversity" has been defined in a variety of ways in the literature, with some using it as a synonym for sentence "interestingness" or "unlikeliness" (Hashimoto et al., 2019), and others considering it a measure of how different two or more sentences are from each other (Vijayakumar et al., 2016; Gimpel et al., 2013). We take the latter approach, and define diversity as the ability of a generative method to create a set of possible outputs that are valid given the input, but vary as widely as possible in terms of word choice, topic, and meaning.

There are a number of reasons why it is desirable to produce a set of diverse candidate outputs for a given input. For example, in collaborative story generation, the system makes suggestions to a user for what they should write next (Clark et al., 2018). In these settings, it would be beneficial to present the user with multiple different ways to continue their story. In image captioning, any one sentence-long caption is probably missing some information about the image. Krause et al. (2017) show how a set of diverse sentence-length image captions can be transformed into an entire paragraph about the image. Lastly, in applications that involve re-ranking candidate sequences, the re-ranking algorithms are more effective when the input sequences are diverse. Re-ranking diverse candidates has been shown to improve results in both open dialog (Li et al., 2016a; Li and Jurafsky, 2016) and machine translation (Gimpel et al., 2013); we have also shown in Section 4.2 how we can leverage this for sentence simplification. Furthermore, in open-ended dialog, the use of re-ranking to personalize a model's responses for each user is a promising research direction (Choudhary et al., 2017).

With these applications in mind, a variety of alternatives and extensions to beam search have been proposed which seek to produce a set of diverse candidate responses instead of a single high likelihood one (Li et al., 2016a; Vijayakumar et al., 2016; Kulikov et al., 2018; Tam et al., 2019). Many of these approaches show marked improvement in diversity over standard beam search across a variety of generative tasks. However, there has been little attempt to compare and evaluate these strategies against each other on a single task.

In this work, we review existing methods for promoting diversity in order to systematically investigate the relationship between diversity and perceived quality of sequences output by conditional language models. In addition to standard beam search and greedy random sampling, we compare several recently proposed modifications to both methods. In addition, we propose the use of over-sampling followed by post-decoding clustering to remove similar sequences, improving upon the methods from Section 4.2.1.[1]

[1]This work was done in collaboration with Daphne Ippolito, Maria Kustikova, João Sedoc, and Chris Callison-Burch. Daphne and I share equal credit for this work, as we came up with the idea together and divided the work evenly.

Method	Description	Method	Description
Random Sampling	Standard decoding mechanism, greedily samples a token from the distribution at each time step.	Random Sampling with Temperature	Before sampling, modify entropy of predicted distribution.
Top-s Random Sampling	Restrict sampling to the s-most likely words in the distribution. (story generation)	Beam Search	Standard decoding mechanism, keeps the top b partial hypotheses at every time step. (MT)
NPAD Beam Search	Add random noise to the hidden state of the decoder at each time step. (machine translation)	Top-g Capping Beam Search	Only consider the top c hypotheses from each parent hypothesis at each time step. (MT, Dialog)
Hamming Diversity Beam Search	Penalize new hypotheses that have many of the same tokens as existing partial hypotheses. (Image Captioning)	Iterative Beam Search	Run beam search several times, preventing later iterations from generating intermediate states already explored. (Dialog)
Clustered Beam Search	Initially consider more hypotheses at each time step, and then cluster similar hypotheses together. (Dialog)	Post-Decoding Clustering (Ours)	Sample a large number of candidates, and then cluster similar outputs together.

Table 16: Brief high-level descriptions of each decoding method we consider. In parentheses we give the applications on which the technique was originally applied (MT refers to machine translation here).

5.1.1. Extensions to Random Sampling

As discussed in Section 2.5.3, one of the primary decoding strategies is known as random sampling, which involves sampling at random from the model's distribution at each time step. Often, a temperature parameter T is added to control the entropy of the distribution before sampling.

$$P(y_t = w_i | y_{<t}, \mathbf{x}) = \frac{\exp(z_{t,i}/T)}{\sum_{j=1}^{V} \exp\left(z_{t,j}/T\right)} \quad \forall i \in \{1, \ldots, V\} \tag{5.1}$$

Choosing a temperature greater than one causes outputs to look increasingly more random, while bringing the temperature less than zero causes sequences to increasingly resemble greedy sampling.

Recently, top-s random sampling has been proposed as an alternative to using temperature. Sampling is restricted to the s most likely tokens at each step Fan et al. (2018); Radford et al. (2019). We find that top-s random sampling's hard-restriction on generating low probability words is more effective at controlling the stochasticity of sampled sequences than sampling with temperature. We also note that subsequent work has proposed nucleus sampling as an additional improvement to the randomly sampling strategies, which involves dynamically choosing s based on the probability distribution Holtzman et al. (2020).[2]

5.1.2. Extensions to Beam Search

In Section 2.5.3, we also discussed beam search, the second commonly-used decoding method which keeps track of b partial hypotheses at each time step. Since beam search only explores a limited portion of the overall search space, it tends to yield multiple variants of the same high-likelihood sequence, sequences that often only differ in punctuation and minor morphological changes (Li and Jurafsky, 2016). Therefore, standard beam search is generally considered to be not ideal for producing diverse outputs. In this section, we discuss a variety of methods that have recently been developed to alleviate redundancy during decoding and generate a wider range of candidate outputs.

Noisy Parallel Approximate Decoding

Introduced by Cho (2016), NPAD is a technique that can be applied to any decoding setting. The main idea is that diversity can be achieved more naturally by taking advantage of the continuous manifold on which neural nets embed language. Instead of encouraging diversity by manipulating the probabilities outputted from the model, diverse outputs are instead produced by adding small amounts of noise to the hidden state of the decoder at each step. The noise is randomly sampled from a normal distribution. The variance is gradually decreased from a starting σ_0 to 0 as decoding progresses (that is $\sigma_t = \frac{\sigma_0}{t}$) under the reasoning that uncertainty is greatest at the beginning of decoding. NPAD can be used in conjunction with any decoding strategy; following the best results from the original paper, we show results using NPAD with beam search. Extensions to NPAD have sought to learn the direction in which to manipulate the hidden states using an arbitrary decoding objective (Gu et al., 2017). Since such objectives can be highly domain-specific, we do not evaluate

[2]Due to monetary constraints, we were unable to re-run our human evaluations to fairly compare this method to all previous ones.

this method.

Top-g Capping

In beam search, it is often the case that one hypothesis h is assigned a much higher probability than all other hypotheses, causing all hypotheses in the next step to have h as their parent. Following Li and Jurafsky (2016) and Li et al. (2016b), we add an additional constraint to standard beam search to encourage the model to choose options from diverse candidates. At each step t, current hypotheses are grouped according to the parental hypothesis they come from. After grouping candidates, only the top g from each grouping are considered. The resulting $b \times g$ candidates are ranked, and the top b are selected as hypotheses for the next beam step.

Hamming Diversity Reward

Vijayakumar et al. (2016) proposes adding an additional diversity-promoting term, θ, to the log-likelihood before reranking. This term measures how different a candidate hypothesis $c^{(i)}_{\leq t}$ is from the partial hypotheses selected in the previous step. Let $\mathcal{H}_{t-1} = \{c^{(1)}_{\leq t-1}, \ldots c^{(b)}_{\leq t-1}\}$ be these partial hypotheses. Then the beam search scoring function for the ith candidate at timestep t becomes:

$$\text{score}(c^{(i)}_{\leq t}) = \sum_{j=1}^{t} \big(\log P(c^{(i)}_j | c^{(i)}_{<j}, \mathbf{x})\big) + \lambda\theta(c^{(i)}_{\leq t}, \mathcal{H}_{t-1}) \tag{5.2}$$

where λ is a tunable hyperparameter. Vijayakumar et al. (2016) attempts a variety of definitions for θ, including embedding diversity and n-gram diversity, but they find that Hamming distance, the number of tokens in the candidate sequence which exist in the previously selected partial hypotheses, is most effective. We take the negative of the Hamming distance as θ.

Iterative Beam Search

In an attempt to improve the size of the search space explored without sacrificing run time, Kulikov et al. (2018) propose an iterative beam search method. Here, beam search is run many times, where the states explored by subsequent beam searches are restricted based on the intermediate states explored by previous iterations. Formally, we can define the set of all partial hypotheses for beam search instance i at time step t as $\mathcal{H}_t^{(i)}$. From here, the search space explored by beam search instance i can be expressed as $S_i = \cup_{t=1}^{T} \mathcal{H}_t^{(i)}$. The i-th beam search is prevented from generating any partial hypothesis that has previously been generated, that is, any hypothesis found in $S_{<i} = \cup_{i'=0}^{i-1} S_{i'}$.

The authors also attempt a soft inclusion criterion, where any states within ϵ Hamming distance from a previously explored state are also excluded. During the experimentation of Kulikov et al. (2018), however, the soft-inclusion was found to not be beneficial; thus, we only restrict exact matches of previous states in our implementation. In practice, this means after the first beam search instance runs as normal, the first step of the second beam search instance will contain the $(b+1)$ to $2b$-most likely starting tokens; this pattern holds for the third beam search instance, and so on.

Clustered Beam Search

Most recently, Tam et al. (2019) proposed a clustering-based beam search method to help condense and remove meaningless responses from chatbots. Specifically, at each decoding step t, this method initially considers the top $2 * b$ candidates. From there, we embed each candidate sequence by computing the averaged GloVe word embeddings (Pennington et al., 2014), and these embeddings are then clustered into c clusters using K-means. Finally, we take the top $\frac{b}{c}$ candidates from each cluster. Note that in the case any clusters have size less than $\frac{b}{c}$, we then include the highest-ranked candidates not found after clustering.

5.1.3. Clustering Post-Decoding (PDC)

In the previous section, we discuss several diversity-promoting methods that can be applied during the decoding process. However, it is also possible to encourage additional diversity post-hoc. On the task of sentence simplification, after decoding using a large-scale diversity-promoting beam search (beam size 100), we can then clustered similar sentences together to further increase the variety of simplifications from which to choose; this was discussed in Section 4.2.1

In this work, we extend this post-decoding clustering idea in three key ways. First, we make use of sentence-level embeddings which leverage the pre-trained language representations from the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019).[3] Second, after clustering, we originally took the sentence closest to the centroid of each cluster as the representative candidate; now, we instead choose the highest-ranked candidate (according to log-likelihood) from each cluster to ensure the best candidates are still selected. Finally, after performing standard k-means clustering, we found that it was often the case that some clusters contained large numbers of good candidates, while others contained very few candidates that are also either ungrammatical or otherwise inferior. Thus, in our implementation, we remove clusters containing two or fewer sentences, and then sample a second candidate from each of the remaining clusters, prioritizing selecting candidates from larger clusters first.

5.1.4. Experimental Setup

We evaluate the decoding strategies described in the previous sections under the following settings. For each of the published beam search algorithms, we choose the hyperparameters that were found to be best in the original publications.

[3]BERT sentence-level embeddings were obtained using https://github.com/hanxiao/bert-as-service.

RS	Random sampling with temp = 0.5, 0.7, 1.0, or 1.0 with top-10 capping.
Standard BS	Standard beam search
Top5Cap BS	Top-g capping with $g = 3$
Iter5 BS	Iterative beam search with 5 iterations
HamDiv0.8 BS	Hamming Diversity with $\lambda = 0.8$
Cluster5 BS	Clustered beam search with 5 clusters
NPAD0.3 BS	Noisy Decoding with $\sigma_0 = 0.3$

For random sampling, we sample 10 outputs, and with beam-search based methods, we use a beam size of 10 to generate 10 outputs. In addition, we show results from oversampling then filtering. We use a beam size of 100 or generate 100 samples through random sampling, and then we select 10 from the 100, either through post-decoding clustering (PDC) or by taking the 10 candidates with highest likelihood. We examine these decoding strategies on two tasks: open ended dialog and image captioning. For each task, we evaluate both the quality and diversity of the 10 outputs from each strategy.

To measure the diversity across the generated candidate sequences for a given input, we report **Dist-k**, the total number of distinct k-grams divided by the total number of produced tokens in all of the candidate responses for a prompt (Li et al., 2016a). We report Dist-2 and Dist-4 averaged over the prompts in the test set.

A limitation of Dist-k is that all k-grams that appear at least once are weighted the same, ignoring the fact that infrequent k-grams contribute more to diversity than frequent ones. Zhang et al. (2018) instead propose an entropy metric, **Ent-k**, defined as:

$$\textit{Ent-k} = \frac{-1}{\sum_{w \in S} F(w)} \sum_{w \in S} F(w) \log \frac{F(w)}{\sum_{w' \in S} F(w')} \tag{5.3}$$

where S is the set of all k-grams that appear in candidate responses for an example, and $F(w)$ denotes the frequency of w in the candidate responses.

Open-ended Dialog Task

In the dialog domain, we use an LSTM-based sequence-to-sequence (Seq2Seq) model implemented in the OpenNMT framework (Klein et al., 2017). We match the model architecture and training data of Baheti et al. (2018). The Seq2Seq model has four layers each in the encoder and decoder, with hidden size 1000, and was trained on a cleaned version of OpenSubtitles (Tiedemann, 2009) to predict the next utterance given the previous one.

Evaluation is performed on 100 prompts from the Cornell Movie Dialog Corpus (Danescu-Niculescu-Mizil and Lee, 2011). These prompts are a subset of the 1000 prompts used in Baheti et al. (2018), and were filtered using item response theory for discriminative power.

We report perplexity (PpL), averaged over *all* the top 10 outputs for each example. This differs from previous work, which computes perplexity over only the top output for each example. For our task, we are interested in the quality of *all* generated responses. Since the quality of open-ended dialog is notoriously difficult to evaluate automatically, we set up a human evaluation task on Amazon Mechanical Turk, where annotators were shown a prompt and 5 potential responses generated by any of our decoding methods. Evaluators were asked to provide binary ratings on Fluency, Adequacy, and Interestingness for each response. Overall, we collected 3 human judgments for each of the top ten responses for each of our decoding methods; in other words, we collected 3,000 judgments per method.

Image Captioning Task

For image captioning, we use a state-of-the-art model introduced in Anderson et al. (2018). We take advantage of Luo (2017)'s open-source implementation and released model parameters trained on MSCOCO (Lin et al., 2014). We evaluate on a test set containing 5000 images.

Method		Fl	Ad	Int	Ppl	Dist1	Dist2	Ent2	Ent4
Reference		0.795	0.732	0.636	–	–	–	–	–
RS 0.7	(rs10)	**0.758**	0.399	**0.388**	35.98	0.63	0.80	4.08	3.84
RS 1.0	(rs10)	0.550	0.303	0.386†	67.99	**0.74**	**0.87**	**4.35**	**4.08**
RS 1.0,top10	(rs10)	0.745†	**0.418**	0.387†	**10.33**	0.60	0.80	4.12	3.91
Standard BS	(bs10)	**0.950**	**0.621**	0.336	**4.01**	0.37	0.45	3.16	3.01
Top3Cap BS	(bs10)	0.942†	0.603	0.346	4.03	0.37	0.46	3.17	3.03
Iter5 BS	(bs10)	0.903	0.520	0.335	5.42	**0.62**	**0.74**	**3.68**	**3.25**
HamDiv0.8 BS	(bs10)	0.923	0.599	0.366†	4.56	0.33	0.37	3.08	3.00
Cluster5 BS	(bs10)	0.936	0.582	**0.381**	4.23	0.39	0.46	3.24	3.06
NPAD0.3 BS	(bs10)	0.942†	0.604†	0.335	4.05	0.36	0.44	3.13	2.99
RS 1.0,top10	(rs100, rank)	**0.922**	**0.548**	0.347	**5.10**	0.52	0.68	3.54	3.18
RS 1.0,top10	(rs100, PDC)	0.852	0.494	**0.372**	6.96	**0.63**	**0.76**	**3.74**	**3.27**
Standard BS	(bs100, rank)	**0.964**	**0.611**	0.332†	**4.01**	0.44	0.61	3.33	3.05
Standard BS	(bs100, PDC)	0.944	0.599	**0.346**	4.42	**0.57**	**0.70**	**3.59**	**3.21**

Table 17: Results on 100 dialog prompts. The first row shows the mean human ratings of the single reference response available for each prompt. The next three rows show results for random sampling, with 10 samples drawn per prompt (rs10). The next six rows are variants of beam search using beam size 10 (bs10). The last four rows use random sampling (rs100) or standard beam search (bs100) to generate 100 outputs, then filter down to 10 outputs either through ranking by log-likelihood or by performing post-decoding clustering (PDC). In each section, the highest value is bolded, and statistical ties are marked †.

We report Semantic Propositional Image Caption Evaluation (SPICE) scores, an automatic evaluation metric that has been shown to correlate well with human judgments of quality (Anderson et al., 2016). SPICE measures how well the semantic scene graph induced by the proposed caption matches one induced by the ground truth. In addition to computing SPICE on the top-scoring caption (SPICE@1), we follow Vijayakumar et al. (2016) in reporting Oracle SPICE@10 scores. This is done to show the upper bound on the potential impact diversity can have. We also compute the mean SPICE score across all of the candidate captions for an image. Unlike SPICE@1 and SPICE@10, this metric shows the overall quality of *all* of the candidate captions, which is useful to know for applications that combine diverse candidate output sequences (Krause et al., 2017).

5.1.5. Results

We report results on dialog systems and image captioning in Tables 17 and 18, respectively. As expected, random sampling-based approaches yield outputs with greater diversity but

Method		SPICE Mean	SPICE @1	SPICE @10	Dist1	Dist2	Ent2	Ent4
RS 0.7	(rs10)	**0.170**	**0.192**	**0.278**	0.31	0.52	3.67	4.00
RS 1.0	(rs10)	0.133	0.167	0.247	**0.44**	**0.71**	**4.17**	**4.26**
RS 1.0,top10	(rs10)	0.159	0.183	0.272	0.33	0.59	3.90	4.17
Standard BS	(bs10)	0.194	0.193	0.283	0.18	0.26	2.94	3.18
Top3Cap BS	(bs10)	**0.195**	**0.196**	0.282	0.17	0.26	2.93	3.17
HamDiv0.8 BS	(bs10)	0.194	0.194	0.282	0.18	0.27	2.98	3.19
Cluster5 BS	(bs10)	0.191	0.194	**0.285**	**0.19**	**0.28**	**3.04**	**3.25**
NPAD0.3 BS	(bs10)	0.191	0.192	0.280	0.18	0.26	2.94	3.17
RS 1.0,top10	(rs100, rank)	**0.182**	**0.188**	**0.284**	0.25	0.41	3.31	3.64
RS 1.0,top10	(rs100, PDC)	0.169	**0.188**	0.282	**0.31**	**0.52**	**3.62**	**3.91**
Standard BS	(bs100, rank)	**0.188**	0.190	0.279	0.20	0.31	3.04	3.32
Standard BS	(bs100, PDC)	0.186	**0.192**	**0.288**	**0.24**	**0.38**	**3.25**	**3.57**

Table 18: Image captioning results for selected random sampling and beam search methods. SPICE@1 measures the SPICE score of the most likely caption. SPICE@10 is the maximum score across the 10 candidates generated by each method. Mean SPICE is the mean score over all 10 candidates. In each section, the best value is bolded.

worse quality than beam search-based approaches. Over-sampling then filtering increases the quality of outputs while still ensuring high diversity. In the following sections, we discuss the diversity-quality tradeoff, and then delve further into the results for each method group.

The Quality-Diversity Tradeoff

The goal of diverse decoding strategies is to generate high-quality candidate sequences which span as much of the space of valid outputs as possible. However, we find there to be a marked trade-off between diversity and quality. This can be seen in Figure 9, where we plot the human-judged quality score for each dialog experiment against our primary diversity descriptive statistics. Fluency and adequacy are both strongly negatively correlated with diversity. While we had expected interestingness to be positively correlated with diversity, the fact that it is not suggests that existing diversity statistics are insufficient for capturing what it means to humans for outcomes to be interesting.

Likewise, in image captioning, the mean SPICE score of the 10 candidate captions (averaged over all examples for each experimental setting) is strongly anti-correlated with diversity, with a Pearson correlation coefficient of -0.83 with the Ent-4 measure and -0.84 with Dist-2. Clearly it remains an open challenge to generate a diverse set of image captions that are all high-quality.

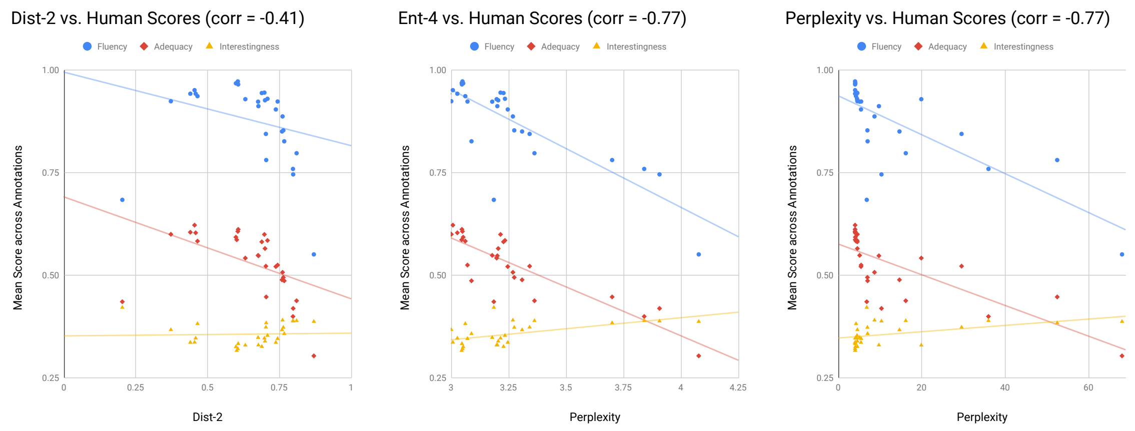


Figure 9: Each decoding strategy is plotted, showing that human-perceived quality is negatively correlated with diversity. The Pearson Correlation coefficients between each statistic and the average of fluency, coherence, and interestingness are shown in parentheses.

When researchers choose to use a diverse decoding strategy, they must decide where on the quality-diversity tradeoff they would like to lie; selecting an optimal method depends strongly on one's tolerance for errors. In machine translation, where mistakes could severely impact coherence, beam search-based methods, which tend to result in better fluency and coherence, but worse diversity might be preferred. In more open-ended applications, where novel text is of greater importance, increased diversity could be worth the fluency and coherency hit. As state-of-the-art models continue to improve, one would hope that the quality cost of encouraging diversity will continue to decrease.

In the interest of reporting a single overall best method for each task, we computed a sum-of-ranks score for each method. For dialog, we ranked the methods each by fluency, coherence, interestingness, and Ent-4, and then took a weighted sum of the four ranks, with 50% of the weight assigned to Ent-4, and 50% distributed evenly among the human evaluation ranks. Overall, clustered beam search and standard BS (beam size 100, PDC) have the best scores, followed by clustered beam search (beam size 10). Similarly, for image

Prompt: Look, nobody knows we did it.

RS 0.5	**Standard BS**	**NPAD0.3 BS**
I don't know what you're talking about.	We've got to get out of here.	I don't think it's a good idea.
What's the matter with you?	What do you mean?	I don't know what to say.
I don't know what it is.	I don't think it's a good idea.	I don't know what's going on.
I don't think so.	I don't know what to say.	I don't know what to do.
He's got to get out of here.	I don't know what's going on.	I don't know what's going on here.

RS 1.0	**Standard BS with PDC**	**Cluster5 BS**
I can't find it.	I don't know!	I don't know why.
They're our ships.	I don't think so.	What do you mean?
It's all right anyone is the right to interfere.	What do you mean?	I don't think so.
We didn't have a plan I engineered a policy.	Why didn't you tell me?	How do you know that?
Same time you pick us up at six and get we.	That's why we're here.	I'll tell you what.

RS 1.0,top10	**RS 1.0,top10 with PDC**	**Top3Cap BS**
I don't know what else to do.	What do you mean?	We've got to get out of here.
It doesn't have to be that way!	I don't think so.	What do you mean?
We're in the air!	That's why I'm here.	I don't think it's a good idea.
I've seen a guy in his place in a it.	It's all right we.	I don't know what to say.
And I'm not we any more.	We've been through this before.	I don't know what's going on.

Prompt: I'd love to drive through New England in the fall.

RS 0.5	**Standard BS**	**Top3Cap BS**
I wish i could have a little talk with you.	I don't know what to say.	If I were you I'd be the first to know.
I've been on the road for three years.	I don't know what to say to you.	If I were you I'd go back to the.
I'd be dead for a long time.	I don't know what to do.	I've got to go to the bathroom.
I'm not a.	I don't know what I'm gonna do.	I'd like to go to the movies.
It's a lovely day.	I don't know if I can do it.	If I were you I'd go back to England.

RS 1.0	**Standard BS with PDC**	**Cluster5 BS**
I'm in love with the world I love.	I don't know what to say.	I'll be back in a few days.
I'd be very happy of you and Kim.	I don't know what to do.	I don't know what to say.
I see and I'd never sleep.	Where are you going?	I don't know what to do.
I'd be dangerous too.	I'll be right back.	I'll be back in a minute.
When mademoiselle would guide the land.	I'd like to go home.	I'll be back in a few minutes.

RS 1.0,top10	**RS 1.0,top10 with PDC**	**NPAD0.3 BS**
I think you're going to be fine.	Why don't you show me how it's done?	I'd like to know what's going on.
I'm a very busy man	No I'll talk to you later.	I don't know what to do.
I don't wanna be in there	I'd like to forget all about it.	I don't know what to do with it.
The of the fall fall fall fall.	You know I was thinking.	I don't know what to say.
And I would've told you to come to Paris.	That's what I do.	I don't know how to thank you.

Table 19: Responses to an example prompt for selected methods. Note that post-decode clustering (PDC) improves the diversity of standard beam search (BS) random sampling.

captioning, we rank the methods by their mean SPICE score and by Ent-4. Summing these ranks, random sampling (temp 1.0, top-10 capping, PDC) came in first. Standard beam search, Hamming Diversity beam search, and Top-g capping beam search (beam size 10) tied for second.

Random Sampling-based Methods

Higher sampling temperatures result in both an increase in diversity in generated responses and a reduction in overall quality. In the dialog domain, evaluators consistently rate the responses sampled with temperature 1.0 to have worse fluency, coherence, and interestingness when those sampled with temperature 0.5. In the image captioning domain, lower temperature improves automatic evaluation metrics for quality while reducing diversity.

For dialog, restricting sampling to the top-10 vocabulary words is a more effective strategy than adjusting temperature for ensuring balance between the quality and diversity of outputs. Top-10 random sampling has the highest fluency, coherence, and interestingness, as well as significantly lower perplexity than other random sampling methods. However, this trend did not extend to image captioning, where top-10 random sampling results in both worse SPICE scores and lower diversity measures than setting the temperature to 0.7. This may be because image captioning is a less ambiguous task than open-ended dialog, leading to a better-trained model that puts more probability mass on high-quality vocabulary words, ameliorating the challenge top-c filtering is designed to eliminate: that of a long tail of low probability vocabulary words taking up a large amount of probability mass.

Beam Search-based Methods

For dialog, clustered beam search (Cluster5 BS) performs the best of all beam search methods in terms of human-judged interestingness. It ties for best with NPAD0.3BS on fluency and ties with Standard BS on coherence. Iterative beam search (Iter5 BS) achieves the greatest diversity, but at the expensive of quality. It has the lowest human-judged coherence among beam search methods; thus, we do not evaluate this method on image captioning. For image captioning, Cluster5 BS has the highest diversity among beam search methods, but this difference is quite small. Cluster5 BS also has the highest SPICE@10 score, indicating it is the best method for generating at least one high quality candidate. However, Top3Cap BS results in the highest mean SPICE score, suggesting it is best at ensuring all outputs are reasonable quality.

Effect of Over-sampling

In our experiments, we explore over-sampling 100 outputs, and then either using post-decoding clustering (PDC) or re-ranking by log-likelihood to filter these 100 down to 10 diverse outputs.

In the dialog domain, this over-sampling approach is a definite win. When over-sampling with random sampling both methods of filtering substantially improve human judgements of fluency and adequacy compared to random sampling only 10 outputs. However, interestingness scores go down, and while the outputs are still more diverse than beam search-based methods, they are less diverse than random sampling without filtering. In the beam search methods that use a beam size of 100 then filter down to 10, human-judged quality is on par with beam size 10 results, but diversity is considerably higher. When comparing the two types of filtering, PDC results in higher interestingness and diversity statistics, while log-likelihood re-ranking improves fluency and adequacy. This again demonstrates the trade-off between quality and diversity.[4]

For image captioning, over-sampling with reranking does not consistently improve quality as it does in the dialog domain. Mean SPICE score is improved for random sampling but not for beam search. SPICE@1 becomes worse for both random sampling and decoding, while SPICE@10 improves for random sampling, and for beam search when PDC is applied. From these results, we can conclude that over-sampling then ranking does not have a sizeable effect, either negative or positive, on quality. Moreover, the diversity of the captions generated by random sampling actually decreases when oversampling. The diversity of beam search-generated captions does improve with over-sampling.

While oversampling does generally improve outcomes on the diversity/quality tradeoff, it is more computationally expensive, particularly with beam search. Running PDC also requires

[4]In the appendix, we show results with every method where we generate 10 samples; generate 100 samples followed by selecting the 10 most likely outputs; and generate 100 samples followed by post-decoding clustering to select 10 outputs.

generating sentence embeddings for every output, which adds additional computation time.

5.2. Summary

In this chapter, we perform an analysis of post-training decoding strategies that attempt to promote diversity in conditional language models. We show how over-sampling outputs and then filtering them down to the desired number is an easy way to increase diversity. Due to the computational expense of running large beam searches, we recommend using random-sampling to over-sample. The relative effectiveness of the various decoding strategies differs for the two tasks we considered, which suggests that the choice of optimal diverse decoding strategy is both task-specific and dependent on one's tolerance for lower quality outputs.

While we have focused on evaluating each decoding strategy under the specifics reported to be the best in the work that first proposed it, further work is necessary to conclude about whether the observed differences in quality and diversity may simply be due to the hyperparameters chosen in each paper. The ability to effectively generate a diverse set of responses while not degrading quality is extremely important in a variety of generation tasks, and is a crucial component to harnessing the power of state-of-the-art generative models.

CHAPTER 6 : Recasting Text Simplification as a Document Retrieval Task

In the previous chapters, we have discussed the state of current lexical and sentence-level simplification systems and proposed ways to extend and improve upon them. While these approaches have proven to be useful in downstream natural language processing tasks, they still have several crucial limitations that prevent these models from being practically useful. First, the models still make many fluency and adequacy errors. These errors are particularly pronounced when models are presented with very long and complex sentences, which makes it difficult to scale up these methods in order to simplify entire documents. In addition, while these models view complexity as a binary Complex to Simple English translation task, complexity is not a binary notion; what is considered complex for one person or group of people may be quite simple for another.

Apart from these issues, when we consider learners who wish to acquire knowledge about a new topic or skill, it is clear that they generally need to understand the totality of a text about this topic. However, often times there are many phrases in a text that cannot be more easily understood just by replacing them with simpler substitutes. As an example, let's consider this excerpt from Wikipedia on *Machine Learning*:

Machine learning (ML) is the scientific study of ***algorithms*** and ***statistical models*** that computer systems use to carry out tasks without explicit instructions....Machine learning algorithms build a mathematical model based on ***sample data***, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.

In order to understand this passage, a reader needs to have prior knowledge of the meaning of phrases such as *algorithms*, *statistical models*, and *sample data.* If the reader does not know what these phrases (or ***Concepts***) mean, they will need to be provided with an explanation. In some cases, it may be enough to provide a definition describing this, as with *sample data* in the above example. However, for many phrases corresponding to relatively complex concepts, such as *algorithms*, it is likely necessary to read longer descriptions written at an appropriate level for the learner.

Due to these issues, rather than focusing on continuing to improve current simplification models, we propose to instead reformulate the problem of text simplification, setting as our goal to help a user learn about a topic with which they may not be familiar. To make the task more tractable, if we start with a document D on a given topic, we can break down the task into three sub-tasks. The first is **Concept Identification**, where we need to identify the concepts within D that are critical for understanding that document. The second task is **Leveled Document Retrieval**, where for each concept c, we need to retrieve a set of documents D' related to c from a *variety* of complexity levels, before re-ranking this set to find documents that are at levels most appropriate for the user.

6.1. Critical Concept Identification

The first step in our proposed approach is to identify the important concepts inside a document. That is, given a document, we want to be able to extract all concepts that are critical to understanding its content. This task can be broken down into two sub-problems: *candidate concept identification*, where we identify all words and multiword expressions that could potentially form a concept; and *candidate score computation*, where we calculate a score for each candidate c that approximates the likelihood of c being a concept.

6.1.1. Methods

Early works on keyword extraction focused on formulating this as a supervised classification task, using features such as the frequency of the most common token, the relative number of characters, and the first relative occurrence of a phrase component (Turney, 2000). Hulth (2003) further showed that incorporating part-of-speech features leads to further improvements in performance. In this work, as our goal is to extract concepts from texts of various domains, we choose to focus mainly on unsupervised approaches.

The first work we compare to is Mihalcea and Tarau (2004). They reformulate the keyword extraction task as a graph-based ranking problem. Specifically, given a document D, they propose to produce a directed graph $G = (V, E)$, where each vertex $v \in V$ represents a single word (i.e. a potential keyword); only nouns and adjectives are considered as potential keywords. A pair of vertices (v_1, v_2) is connected by an edge $e \in E$ if the words associated with these vertices co-occur within a window of N words, where N is a hyperparameter to be tuned through experimentation. The score $S(v_i)$ of each vertex v_i is initialized to 1, and then updated during each iteration of the algorithm as follows:

$$S(v_i) = (1 - d) + d * \sum_{j \in In(v_i)} \frac{1}{Out(v_j)} * S(v_j) \tag{6.1}$$

$In(v_i)$ represents the set of vertices that have an edge pointing towards a vertex v_i, $Out(v_i)$ represents the set of vertices pointed to from v_i, and d is a dampening parameter, set to 0.85 (Mihalcea and Tarau, 2004). The top t keywords, as ranked after the algorithm has converged, are passed through a post-processing step which combines adjacent adjectives and nouns if both are identified as keywords.

Subsequent work has attempted to leverage multi-word expressions more directly, as single words can be used in many different contexts; a notable example of this is Rapid Automatic Keyword Extraction, or RAKE (Rose et al., 2010). This approach generates a set of stopwords and phrase delimiters, and leverages this to partition a document into a set of candidate keywords. RAKE creates a graph of word co-occurrences G, meaning that two words have an edge between them when they are found in the same candidate. A score is then computed for each word w based on its frequency ($freq(w)$), the degree of its corresponding vertex in G ($deg(w)$), and the ratio between its degree and its frequency ($\frac{deg(w)}{freq(w)}$); finally, an overall score is computed for each candidate by taking the sum of its word scores. To account for concepts that contain stopwords (e.g. *axis of evil*) candidates that are found adjacent and in the same order multiple times within a document are combined. RAKE returns the top t percentage of candidates as true keywords; in this work, t is set to 0.33.

Similar to RAKE, YAKE (Yet Another Keyword Extractor) is a multilingual unsupervised algorithm that uses features extracted from a single document and only requires a static list of stopwords as external data (Campos et al., 2020). After applying a similar pre-processing step as the RAKE algorithm, YAKE computes five features for each word: *casing* (i.e. the capitalization of a word); *word positional* (i.e. the relative position of the word within the document); *word frequency* (i.e. the frequency of a word within the document); *word relatedness to context* (i.e. the number of unique word that occur close to a term); and *Word DifSentence* (i.e. how often a word appears in distinct sentences). Each word w is then assigned a score $S(w)$, and each keyword k is assigned a score as follows (here, $TF(k)$ represents the term frequency of k):

$$S(w) = \frac{\prod_{w \in k} S(w)}{TF(k) * (1 + \sum_{w \in k} S(w))} \tag{6.2}$$

Note that unlike RAKE, there is an extra normalization step, which serves to avoid artificially inflating the scores of longer n-grams. Finally, YAKE outputs a list of keywords sorted by the score, where a lower score indicates a more important phrase.

In our work, we also consider several methods that extend previous approaches. In order to create our own method, we first need to extract all candidate concepts from a document. This is a non-trivial task, as there is potentially an exponential number of possible phrases that can be extracted from a single document. Previous methods often focus on n-gram-based approaches for candidate identification, which allow the models to be relatively quick and scalable; however, these can result in poor phrasal boundaries, sometimes mixing verbs and nouns in the same phrase. In this work, what we only consider noun phrases "candidate concepts". We thus choose to use the candidates identified by YAKE for all of our extended methods, as true candidates must be contained within a single chunk within a sentence.

Several recent works have naturally leveraged Wikipedia for concept identification and entity linking (Ratinov et al., 2011; Upadhyay et al., 2018), as it contains a millions of articles about specific topics or entities. To organize these documents, Wikipedia labels each document with a set of categories. These categories are also organized into a sort of hierarchy; for example, the subcategories under "Computer Science" include "Artificial Intelligence" and "Computer Architecture". This allows us to extract a large set of concepts relevant to a target domain. Most of our proposed methods leverage these categories. One issue with extracting concepts this way is that this "hierarchy" actually contains many cycles, i.e. following this hierarchy all the way to the bottom results in many irrelevant categories. For example, attempting to build a complete hierarchy of concepts starting under "Computer Science" eventually results in the inclusion of categories such as "flower", which is clearly an out-of-domain concept. This is an issue we need to keep in mind as we use this large-scale general corpus. To get around this, we extracted a hierarchy of concepts that were at most eight steps beneath "Computer Science", our target domain for these experiments. This process results in 106,359 categories. If we also include the titles of all Wikipedia pages labeled with these categories, this results in 792,555 total Computer Science-related concepts. We leverage these as baseline methods (**Wiki-Cat** and **Wiki-All**) by performing a simple dictionary lookup or each candidate concept.

Inspired by recent work that applies BERT-based models to keyphrase extraction (Sharma and Li, 2019), we consider several unsupervised approaches leveraging pre-trained language models. In our first approach, given a document text t and each candidate concept $c \in C$ identified within t, we generate a score for c by calculating the cosine similarity between the SBERT embeddings of t and c; we refer to this method as **DocSim**.[1] Note that we use SBERT over BERT here as SBERT embeddings were created to make similarity comparisons more meaningful (Reimers and Gurevych, 2019).

One potential issue with this first approach is that the number of words in the entire

[1]This can be considered a variation of KeyBERT https://github.com/MaartenGr/KeyBERT.

document is much larger than the number of words in a single candidate concept, thus the similarity between embeddings may not be the most meaningful. To get around this issue, we also consider comparing each candidate $c \in C$ by computing its SBERT embedding similarity to a document's title; we refer to this method as **TitleSim**.

While using the title significantly cuts down on the number of words, it is important to note that the title often contains stopwords such as "the" and "of" that could make this process less accurate, resulting in some relevant concepts still being excluded. Thus, we also extend the method of Tsai et al. (2013), which compares each candidate concept to concepts in a small manually collected seed dictionary. In our experiment, to create our seed dictionary, we take the top-k most relevant concepts based on their SBERT embedding similarity to the document title; we choose $k = 2$ through experimentation. For each candidate concept $c \in C$ we then compute the maximal embedding similarity between c and the seed concepts.

6.1.2. Data Collection

For evaluation, previous work has mainly leveraged corpora consisting of abstracts from technical articles. Mihalcea and Tarau (2004) and Rose et al. (2010) both evaluate TextRank and RAKE, respectively, on the 500 test texts from the Inspec dataset, which consist of abstracts from journal papers in the Computer Science and Information Technology domain. Similarly, Tsai et al. (2013) evaluate on a set of abstracts taken from the ACL Anthology annotated with concepts identified as either *focus*, an article's main contribution; *technique*, a method or tool used in the article; and *domain*, an article's application domain (Gupta and Manning, 2011).

In our work, while technical abstracts are helpful to consider, we also want to determine the efficacy of these models on web-based articles as well. Thus, we extract a set of 41 web-based articles from the computer science domain, and train two annotators to manually annotate concepts for each document. These articles are from the following sources:

- Seven abstracts from the journal *Data Science*.

- Nine articles from *Edureka* and *Intellipaat*, two introductory computer science websites.
- Eleven articles from *python-course.eu* and *python.swaroopch.com*, websites with introductory Python articles.
- Seven articles from *GeeksforGeeks*, a website with a variety of technical and coding computer science articles and problems.
- Five articles from *Wikipedia*.
- Two articles from *marksheet.io*, a website focused on introductory HTML articles.

The guidelines given to annotators are described in Appendix A.1. After the initial annotation process, the annotators were asked to perform an additional adjudication for each phrase that was disagreed upon. In order to determine the level of agreement between annotators, we compute Cohen's Kappa coefficient. The general formula for Cohen's Kappa is below; here, p_e represents the expected probability of an event occurring, and p_o represents the observed probability that the event occurred:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{6.3}$$

To compute this, we can formulate our problem as follows: for each candidate concept $c \in C$, each annotator gave either a positive label (i.e. c is a critical concept) or a negative label (c is not a critical concept). We can compute *Equal Kappa*, where we weight agreement on positive and negative labels equally. We make the assumption that the expected probability of identifying c as a concept is 0.5.

$$p_e = P(\textit{Binary Agreement}) = 0.5 \tag{6.4}$$

$$p_o = \frac{\#\ \textit{Total agreed upon annotations}}{\#\ \textit{Total noun phrases}} \tag{6.5}$$

To compute the total number of noun phrases, we generate a full constituency parse of each

	Equal Kappa	Positive Kappa
Pre-Adjudication	0.806	0.272
Post-Adjudication	0.841	0.488

Table 20: Annotation agreement metrics before and after the secondary adjudication step.

sentence in each article, and extract all non-nested noun phrases.[2] Note that these articles are all from the computer science domain, so they often contain mathematical equations and/or coding snippets. To reduce confusion for the constituency parser (as well as any concept extraction models later on), we use html tags to automatically remove code snippets from our test articles, and manually clean the remaining.

While Equal Kappa is reasonable at first glance, the main issue with this metric is that the expected probability in our task is really not 0.5, as there are significantly more noun phrases than there are identified concepts. Thus, we propose a second agreement metric, called *Positive Kappa*, where we only focus on agreement for phrases identified as concepts.

$$p_e = \frac{\#\ Annotator\ 1\ concepts + \#\ Annotator\ 2\ concepts}{2 * \#\ Total\ noun\ phrases} \tag{6.6}$$

$$p_o = \frac{\#\ Total\ agreed\ upon\ concepts}{\#\ Total\ identified\ concepts} \tag{6.7}$$

To use an example, if there are 100 total noun phrases, and Annotator 1 identified 10 concepts, while Annotator 2 identified 15 concepts, the expected agreement, i.e. if these were identified at random, would be $\frac{10+15}{2*100} = 0.125$. The agreement metrics before and after adjudication are shown in Table 20. As expected, having our annotators do a second pass where they focus on cases of disagreement results in a significant increase in positive Kappa. Finally, to generate a single set of gold concepts for each document, we only use concepts that were identified by both annotators; in total, this results in 657 concepts identified in the 41 documents of our dataset.

[2]We generate the constituency parse using Spacy's python implementation of the Berkeley Neural Parser Kitaev and Klein (2018).

6.1.3. Experiments

We compare different standard keyphrase extraction methods with additional baseline and BERT-based approaches. For several methods, we use different ranking thresholds to filter out different numbers of candidate concepts. For example, a ranking threshold of 1 means that we will use all the candidate concepts, effectively ignoring the model ranking; on the other hand, a ranking threshold of 0.33 means that we will take only the top 33% candidates in the ranking produced by some method.

We compare the following methods:

- Mapping to all computer science Wikipedia categories (**Wiki-Cat**), starting with all candidate concepts identified by YAKE (Campos et al., 2020).
- Mapping to all computer science Wikipedia categories and article titles (**Wiki-All**), starting with all candidate concepts identified by YAKE.
- **TextRank** (Mihalcea and Tarau, 2004), using ranking thresholds of 1 and 0.33.[3]
- **RAKE** (Rose et al., 2010), using ranking thresholds of 1, 0.33, and 0.2.[4]
- **YAKE** (Campos et al., 2020), using ranking thresholds of 1, 0.33, 0.2, and 0.1.
- **KeyBERT**, a simple BERT-based approach based on Sharma and Li (2019), using ranking thresholds of 1 and 0.33.[5]
- **DocSim**, an SBERT-based approach comparing candidate concepts to the embedding of the entire document; starting with candidate concepts identified by YAKE, using a ranking threshold of 0.33.
- **TitleSim**, an SBERT-based approach comparing candidate concepts to a document's title; starting with candidate concepts identified by YAKE, using a ranking threshold of 0.33.
- **SeedSim**, an SBERT-based approach first comparing candidate concepts to a document's title, identifying k seed concepts, before comparing candidates to the seed con-

[3]We use the Python implementation of TextRank (Nathan, 2016) integrated into the SpaCy framework (www.spacy.io).

[4]We use the Python implementation of RAKE at https://github.com/aneesha/RAKE.

[5]We use the Python implementation found at https://github.com/MaartenGr/KeyBERT.

Method	Threshold	Precision	Recall	F-Score
Wiki-Cat	–	0.200	0.143	0.167
Wiki-All	–	0.183	0.311	0.231
TextRank	1	0.097	0.469	0.160
	0.33	0.199	0.310	0.242
RAKE	1.0	0.119	0.766	0.212
	0.33	0.168	0.363	0.230
	0.2	0.189	0.250	0.215
YAKE	1.0	0.075	**0.943**	0.139
	0.33	0.177	0.401	0.246
	0.2	0.219	0.303	**0.254**
	0.1	**0.268**	0.189	0.221
KeyBERT	1.0	0.027	0.898	0.053
	0.33	0.024	0.156	0.042
TitleSim	0.33	0.149	0.272	0.183
SeedSim	0.33	0.132	0.272	0.177
DocSim	0.33	0.137	0.175	0.153

Table 21: Comparison of concept identification methods on our manually annotated test set.

cepts; starting with candidate concepts identified by YAKE, using a ranking threshold of 0.33.

The results are shown in Table 21. As we can see, surprisingly all models that leverage large pre-trained language models perform quite poorly on our evaluation set. Regarding KeyBERT, this is likely due to the n-gram-based approach to identifying candidate concepts being quite noisy. In addition, comparing candidate concept embeddings to that of the entire document likely is an inappropriate comparison, due to the sheer difference in number of words. Regarding our methods relying on the title, this seems like a reasonable approach when the title is a technical concept or a combination of concepts related to the document; however, in our dataset this seems to often not be the case. For example, one of the Wikipedia articles is entitled *TenTen Corpus Family*; the title of a corpus followed by the word *family* is likely not a meaningful phrase to compare to in embedding space. Looking at the results overall, YAKE slightly outperforms all other baselines, though our simple dictionary lookup using Wikipedia computer science topics is surprisingly competitive.

For a given complex document, our task is to retrieve documents on the same topic but written at different complexity levels. This means that our document-level representation should be able to capture semantic similarity across levels. A challenge for this task is that the vocabulary used in a simple document is generally quite different from that used in complex documents, making it difficult to use standard TF-IDF-based retrieval methods. Therefore, we instead experiment with several different contextual embedding representations.

We leverage recent advancements in BERT-based language models (Devlin et al., 2019; Reimers and Gurevych, 2019), and explore their ability to capture both semantic (topic) similarity and text complexity. As BERT-based models can process up to 512 WordPiece tokens (henceforth tokens) at a time, it is not possible to generate a single embedding for an entire document D at once. Thus, we split D into a set of paragraphs $\mathbf{P}$, denoted in D through new line markers, and combine them into text segments $S_i \in D$ of length up to 512 tokens. We generate embeddings for these text segments $\mathbf{E}_{S_i}$ and compute their average weighted by their length (l_{S_i}). Formally, we compute each dimension $e_j \in \mathbf{E}_D$ of our document-level embedding as follows:

$$e_j = \frac{\sum_{S_i} e_{S_i,j} * l_{S_i}}{\sum_{S_i} l_{S_i}}$$

Given an input document D_s and a set of documents D_t in a corpus $\mathbf{C}$, we rank the documents in C based on the cosine similarity of their embeddings E_{D_t} to that of the input, E_{D_s}.

$$sim(D_s, D_t) = cos\_sim(E_{D_s}, E_{D_t})$$

In this experiment, we use both BERT (Devlin et al., 2019) and Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) pre-trained embeddings. As SBERT generates sentence-level embeddings that are more semantically meaningful (Reimers and Gurevych, 2019), we expect that SBERT will outperform BERT. Apart from directly using static pre-trained embeddings, we also attempt a fine-tuned approach. To do so, we adapt the Siamese BERT-network architecture introduced by SBERT (Reimers and Gurevych, 2019) with the goal of learning that texts on the same topic at different complexity levels are similar, while texts about different topics at the same complexity level are not similar. We do this by fine-tuning a Siamese BERT network on a binary classification task. The positive examples (labeled 1) consist of an aligned text pair (T_{ij}, $T_{ij'}$) containing similar content, where $T_{ij'}$ is at a lower complexity level than T_{ij}. The negative examples (labeled 0) consist of a non-aligned text pair (different content) written at the same complexity level (T_{ij}, T_{kj}).

6.2.1. Document Retrieval Across Complexity Levels

In our initial retrieval experiment, we leverage the original 1,882 aligned documents from Newsela (Xu et al., 2015). Level 0 (or L0) in Newsela represents the lowest complexity level, and level 4 (L4) the highest. For this experiment, we split the corpus in two groups of 941 documents sets, where each set consists of five aligned articles. We use the first group to fine-tune our Siamese BERT-based network, and save the second group for evaluation of our retrieval approach.

We use each L4 Newsela document D_{i4} as our input and generate an embedding representation $E_{D_{i4}}$. We then generate embeddings $E_{D_{kj}}$ for all other documents D_{kj} in Newsela. These include the four re-written versions D_{ij} of each input document, where $0 \leq j \leq 3$ (j indicates the complexity level of the simplified version aligned to the original document D_{i4}). We experiment with different types of representations.

Pre-trained Contextualized Representations We use pre-trained BERT and SBERT embeddings generated for text segments of length up to 512 BPEs, combined into a single document-level representation through our length-based weighting procedure discussed in the previous section.

Fine-tuned Representation We fine-tune a Siamese BERT-network to better account for differences in complexity. Fine-tuning is done in two ways.

- **BERT-Sent**: Fine-tuning on sentence pairs (s_1, s_2), analogous to the original SBERT methodology (Reimers and Gurevych, 2019). For positive examples, we use aligned Newsela sentences, where s_1 is more complex than s_2.[6] To generate negative examples, we form a sentence pair that consists of s_1 from a positive example, and a sentence s_3 adjacent to s_1 in the Newsela document where it occurs; we choose these examples because s_3 is at a similar complexity level and about a similar topic, but does not share the same meaning.

- **BERT-Doc**: Fine-tuning on document pairs (D_1, D_2). As positive examples, we use pairs of aligned documents from Newsela (D_{i4}, D_{ij}), where the first document is always at L4 and j can range from 0 to 3. For negative examples, we pair D_{i4} from the positive examples with a different L4 document D_{k4}; these are chosen because D_{k4} is at a similar complexity level, but is not about the same topic.

In both settings, we train the models for one epoch. After training, we use these fine-tuned models to generate document-level embedding representations for use in our retrieval task.

After generating the pre-trained or fine-tuned embeddings, we compute the similarity between D_{i4} and every other document D_{kj}. We report the rank of each aligned document D_{ij}, where $0 \leq j \leq 3$. We follow two approaches for reporting the final ranking, each capturing a different aspect of the overall effectiveness of the retrieval method.

- **Keep all aligned documents (KAAD)**: In this setting, the four documents aligned with D_{i4} effectively compete against each other in the ranking.

[6]Newsela sentence pairs were aligned by Jiang et al. (2020)

Embedding Method	**Target Aligned Document Complexity Level (Newsela only)**							
	L3		**L2**		**L1**		**L0**	
	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc
BERT	**0.998**	**0.998**	**0.996**	**0.993**	0.818	0.772	0.403	0.339
SBERT	0.988	0.983	0.967	0.957	**0.905**	**0.872**	**0.724**	**0.674**
BERT-Sent	0.816	0.775	0.513	0.452	0.119	0.089	0.014	0.009
BERT-Doc	0.666	0.579	0.248	0.173	0.054	0.028	0.016	0.007

Table 22: Ranking results for retrieving aligned Newsela documents at four complexity levels, given a source document at L4 and using the ROAD configuration. We report results using both static BERT and SBERT embeddings and fine-tuned approaches (BERT-Sent and BERT-doc).

- **Remove other aligned documents (ROAD)**: In this configuration, we only keeps a single aligned document in the ranking at a time.

The ROAD approach allows for a clean comparison of a single aligned document with 4,500 non-aligned documents, so we initially report results for this configuration. However, in the KAAD scenario there will be multiple relevant documents at different complexity levels, so this is likely more practically relevant for our problem. We report these results in a later section.

We use two metrics to measure how well our embedding-based methods retrieve aligned documents: Mean Reciprocal Rank (MRR) and Top-1 Accuracy (Top1Acc). MRR is used here because simply computing the average rank would inflate the influence of low-ranked documents. Top1Acc has the advantage of being more easily interpretable, and also in downstream applications it is highly important that the most relevant document be ranked first.

The results of this experiment are reported in Table 22. BERT and SBERT are both highly successful in retrieving the aligned L3 and L2 articles, with BERT performing slightly better. However, SBERT strongly outperforms BERT when retrieving L1 and L0 articles. This suggests that SBERT is able to capture meaning similarity between texts with higher surface-level linguistic variation better than BERT.

Interestingly, accounting for complexity information through fine-tuning substantially re-

Embedding Method	**Target Aligned Document Complexity Level (Newsela + NYT Distractors)**							
	L3		**L2**		**L1**		**L0**	
	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc
BERT	**0.996**	**0.995**	**0.962**	**0.943**	0.548	0.479	0.169	0.128
SBERT	0.917	0.892	0.812	0.772	**0.613**	**0.550**	**0.336**	**0.301**

Table 23: Ranking results for retrieving aligned Newsela documents at four complexity levels, including the 1.8 million NYT articles as distractors in the ROAD configuration. The source Newsela documents are at L4.

duces performance, especially when retrieving L1 and L0 documents. This suggests that fine-tuning a BERT-based model to better understand when texts discuss similar topics at disparate complexity levels comes at the cost of affecting the overall embedding quality. For the document-level fine-tuning experiment, this may be due to insufficient data; Newsela only contains several thousand documents, while the Siamese-BERT network was originally trained on over one million NLI sentence pairs (Reimers and Gurevych, 2019).

6.2.2. Including Additional Distractors

While the results using static BERT and SBERT representations from Section 6.2.1 are promising, this can be thought of as a relatively “clean room” experiment carried out on a small aligned corpus. In this controlled setting, we are only using 4,595 (919*5) news articles as potential “distractor” documents. In a more realistic scenario, information retrieval is a much harder task because of the very high number of unrelated documents. To simulate this scenario, we augment Newsela with a large set of articles from the New York Times (NYT).[7] Specifically, we include 1,828,842 NYT articles from 1991 to 2018, each containing at least five paragraphs, and treat these as distractor documents. Note that we are making the relatively strong assumption that all NYT articles are not related to the Newsela articles, which may not be the case given the size of the corpus.

In this experiment, we only use static representations generated by pre-trained BERT and SBERT, since fine-tuning was shown to lower performance. Computing all pairwise cosine

[7]The NYT articles were extracted from Penn’s library API by Sihao Chen.

similarities would be computationally expensive and time consuming, and certainly not ideal in a practical setting. We alleviate this issue by leveraging an approximate k-nearest neighbors (k-NN) approach, implemented in the ANNOY toolkit (Bernhardsson, 2018). Approximate k-NN is significantly faster than exact k-NN (Patel et al., 2018). After creating an ANNOY index containing all document embeddings, we use each L4 Newsela document D_{i4} as input, and find the documents that are most similar to it according to approximate k-NN.

The results of this experiment are given in Table 23. As in the previous experiment, we report MRR and Top-1 Accuracy for retrieving the four aligned Newsela articles. Similar to the results from Section 6.2.1, our BERT-based and SBERT-based retrieval approaches are both able to effectively identify aligned documents at the L2 and L3 levels, which are closer to L4. We see that BERT further outperforms SBERT in this task. However, at the L1 and L0 levels, we see that SBERT outperforms BERT, highlighting again that SBERT is stronger at capturing the similarity of documents with diverse complexities. The performance of retrieving L1 and L0 documents is significantly lower in this experiment. This is not surprising, since adding in 1.8 million extra distractors makes this a much more challenging retrieval task.

6.3. Error Analysis

To determine why we need a separate re-ranking model, it is worth taking a closer look at examples of aligned documents that were initially ranked very low by our BERT-based retrieval methods. For this analysis, we consider aligned versions that were ranked outside of the top 500 most relevant documents; our BERT-based method made 101 errors of this type, while our SBERT-based method made 11 such errors. Qualitatively, we observe that the "missed" aligned document often only contained a fraction of the original context. We suspect that an important aspect of simplification is that significantly simplifying text can result in changing the original meaning. An abbreviated example of this phenomenon are shown in the Table 24, while more examples can be found in Appendix A.2.

Source L4 Document
Facebook Chief Executive Mark Zuckerberg announced Tuesday that he plans to eventually donate 99 percent of the Facebook stock owned by him and his wife, Priscilla Chan, shares that are worth about $45 billion today. That amount would make it one of the largest philanthropic commitments ever.
Zuckerberg said in an open letter that the birth last week of their first child, a daughter named Max, was the motivation to dedicate their considerable wealth to charitable causes now. They wrote that they did not want to wait to "advance human potential and promote equality for all children."

Missed Aligned L0 Document
Mark Zuckerberg runs Facebook. He helped start the company when he was a in college.
Facebook is used by billions of people around the world. It has made Mark a great deal of money. He is very, very rich.
Mark had some big news his week. He said he and his wife will give away much of their money. His wife's name is Priscilla Chan. She is a doctor.
The two promised to give away around $45 billion. The money will be used to help make schools better.
News Was Delivered On Facebook
Mark said this on Facebook. It was in a letter to his first child. Her name is Maxima Chan Zuckerberg.

Table 24: Example of an aligned document pair missed by our BERT-based model, titled *zuckerberg-donation*.

To further explore this phenomenon, we consider the number of tokens and the number of unique noun phrases (NPs) as proxies for content quantity and quality in Newsela documents.

For each "missed" aligned document, we extract unique NPs from the following articles:

- **Source**: The source L4 document D_{i4}.
- **Best Aligned**: The highest ranked (i.e. most similar) aligned document to D_{i4}; this is almost always D_{i3}.
- **Best Non-aligned**: The highest ranked non-aligned document to D_{i4}.
- **Missed aligned**: The aligned document that was erroneously ranked very low; this is almost always D_{i0}.

In Table 25, we report the average number of tokens in the original Source document compared to that of the Missed Aligned article. While some length difference is expected across Newsela complexity levels, as reported in Section 2.3.2, the length difference when our models make a mistake is substantially larger than expected, especially when using SBERT. For each of these articles, in addition to their length we compute the average number of

Error Statistics	**BERT**	**SBERT**
Source # Tokens	1,119	1,495
Missed Aligned # Tokens	463	432
Source Noun Phrases	275	324
Best Aligned Noun Phrases	241	224
Best Non-aligned Noun Phrases	302	303
Missed Aligned Noun Phrases	71	67
Source/Best Aligned Overlap	0.613	0.471
Source/Best Non-Aligned Overlap	0.015	0.009
Source/Missed Aligned Overlap	0.051	0.055

Table 25: Statistics from errors made by BERT and SBERT in the initial document retrieval experiments. We report the average number of unique noun phrases for the Source L4 article, the Best Aligned article, the Best Non-Aligned article, and the Missed Aligned article, along with the fraction of noun phrase overlap with the source article.

unique NPs, as well as the degree of NP overlap between Source and the other 3 articles. The number of noun phrases in the Missed Aligned articles is significantly lower than that in the other articles, while the overlap between Source and Missed Aligned is quite low. The sheer size of the decreases, both in terms of length and number of unique noun phrases, and the lack of noun phrase overlap further supports the suggestion that some documents at lower complexity levels have substantially less content than their aligned version at higher levels.

6.4. Document-Level Complexity Prediction

As mentioned above, in most practical retrieval settings (e.g., search), it is vital for the most appropriate document to be ranked first. However, we have shown that methods that rely on contextualized embeddings struggle to highly rank aligned documents at more disparate levels, and our initial attempts to fine-tune embeddings for capturing complexity resulted in even lower performance. In this section, we propose to train a separate model to predict document-level complexity, which we can then leverage in order to re-rank the originally retrieved results and boost documents at the appropriate complexity level.

6.4.1. Methodology

Most recent work on complexity prediction has been at the word level (Shardlow, 2013a), and at the sentence level in the context of re-ranking sentence simplification system outputs (Zhang and Lapata, 2017). In this work, we are instead interested in predicting document-level complexity. We fine-tune BERT for complexity prediction using examples that contain a single text segment and a label corresponding to its complexity level. Following the original fine-tuning procedure from Devlin et al. (2019), we add a single dense layer of neurons on top of the original pre-trained BERT model, and re-train the full architecture.

To output a single document-level prediction, we break down documents into text segments of up to 512 tokens, and fine-tune BERT on these segments. At test time, we get a single predicted complexity p_D for a document D by generating predictions p_{S_i} for each text segment $S_i \in D$, and selecting the mode prediction weighted by the length of each segment l_{S_i}. Formally, we compute this as follows:

$$w_c = \sum_{S_i} (l_{S_i} \mid p_{S_i} = c) \text{ , where } c \in [0, 4]$$

$$p_D = \arg\max_c (w_c)$$

6.4.2. In-Domain Experiment and Results

For this experiment, we again use Newsela. We fine-tune on the same 941 article sets used for fine-tuning in Section 6.2.1, using 90% of Newsela articles as training data, and 10% as validation data. The other 941 Newsela articles (used for evaluating our retrieval approach) are held out as test data.

To generate training examples for fine-tuning BERT, we split each Newsela document D into multiple text segments as before, and label each of them with the same complexity level. We fine-tune for two epochs, as longer training results in overfitting.

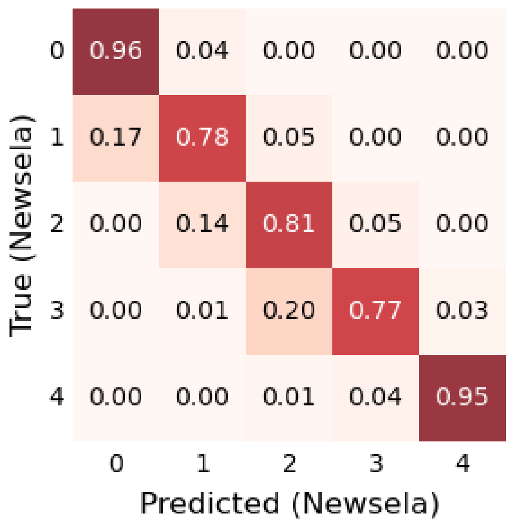


Figure 10: Confusion matrix of predicted complexity levels after fine-tuning BERT on Newsela vs. true complexity levels.

The fine-tuned BERT model has an overall **accuracy of 0.764** on individual text segments, and a document-level **accuracy of 0.844** in our held-out test set. We also report the accuracy of our complexity prediction model for each complexity level in the confusion matrix in Figure 10. L0 and L4 documents were almost always correctly classified, while the complexity of L1, L2, and L3 documents was less accurately predicted. However, when the model made a mistake, it was in most cases only one level off. From these results, we can conclude that fine-tuning BERT is a very effective method for predicting in-domain document-level complexity.

6.4.3. Cross-Domain Experiment and Results

While our in-domain results are quite promising, in order for this approach to be practically useful, our model must be able to perform well across different domains. To simulate this setting, we leverage additional datasets. The first the Weebit corpus which consists of a combination of two corpora created for different age groups (Vajjala and Meurers, 2012): the WeeklyReader corpus [8] and the BBC-Bitesize website[9]. The final Weebit corpus consists of

[8]WeeklyReader has merged with its parent company ScholasticNews, so these articles can now be found at https://scholasticnews.scholastic.com

[9]http://www.bbc.co.uk/bitesize

Grade	Age	Newsela		Weebit		Choosito	
		Level	# Articles	Level	# Articles	Level	# Articles
2	7-8		35	Level 2	629	Early	
3	8-9		124	Level 3	801	Early	4,044
4	9-10		1,064	Level 4	814	Early	
5	10-11		864	KS3		Emerging	
6	11-12		680	KS3	644	Emerging	14,283
7	12-13	varies	685	KS3		Emerging	
8	13-14		743	KS3		Fluent	
9	14-15		338	GCSE	3,500	Fluent	13,944
10	15-16		22	GCSE		Fluent	
11	16-17		2	N/A	N/A	Advanced	6,368
12	17-18		1,093	N/A	N/A	Advanced	

Table 26: Comparison of reading levels across the Newsela, Weebit, and Choosito corpora.

the following levels from these corpora: Level 2 (corresponding to grade level 2 and children ages 7-8); Level 3 (grade 3, ages 8-9); and Level 4 (grade 4, ages 9-10); KS3 (ages 11-14) and GCSE (ages 14-16).

We also introduce a new corpus which consists of 38,639 articles extracted from the Choosito website (henceforth the Choosito corpus) manually labeled at four reading levels: Early (grade levels 2-4); Emerging (grades 5-7); Fluent (grades 8-10); and Advanced (grades 11-12).[10] During corpus creation, any article that contained a sentence with more than 100 words was removed. This was done because we found that these sentences generally consisted of lists of items, which made the text extracted from these articles difficult to parse.

These corpora align documents with school grade levels. The Newsela corpus does initially use grade level to ensure that the five aligned documents are written at 5 different complexity levels, but these do not always correspond to the same grade levels. The 1,130 aligned documents in the first version of Newsela (Newsela V1) still contain this grade level information. However, this is not the case for Newsela V2, which is the most commonly used version in simplification studies. In order to fairly compare across these corpora, we used Newsela V1 along with its grade level information, and re-organized Newsela levels

[10]The Choosito corpus was compiled by Jaime Rojas at Choosito, Inc., and was shared with us for this work.)

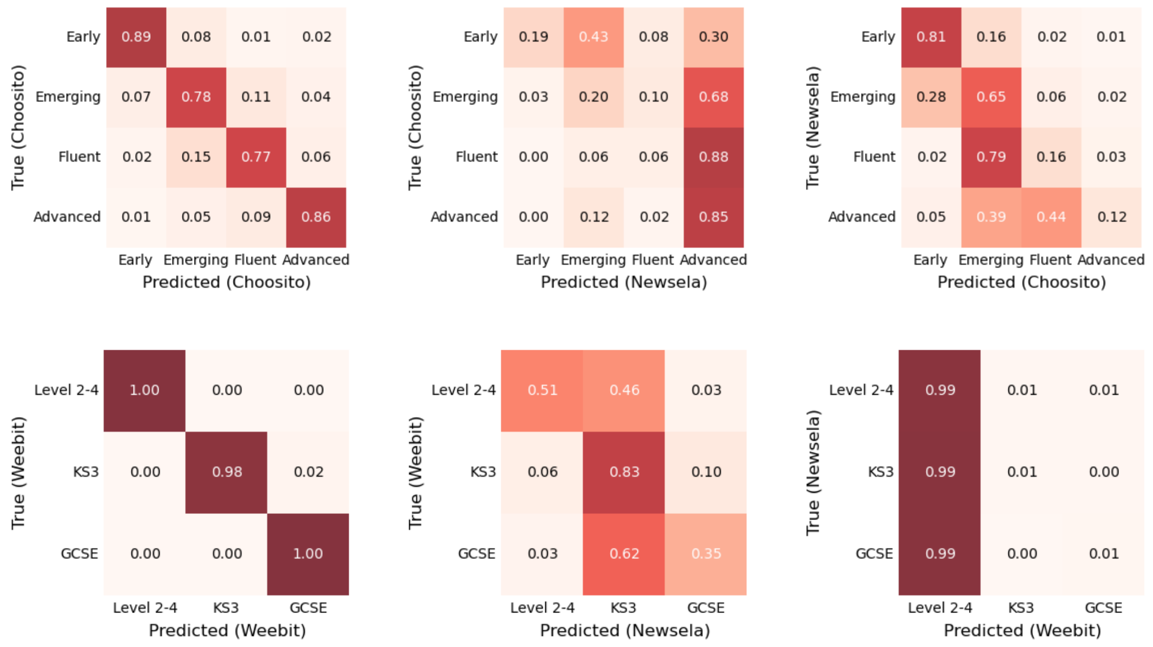


Figure 11: Confusion matrix for in-domain and out-of-domain performance across reading levels. The first row shows the performance of training/testing on the Choosito corpus, training on Newsela/testing on Choosito, and finally training on Choosito/testing on Newsela. The second row shows the performance of training/testing on the Weebit corpus, training on Newsela/testing on Weebit, and finally training on Weebit/testing on Newsela. Note that when using Newsela, we modify the reading levels to match the second corpus.

depending on the corpus used for training/testing. The distribution of documents across grade levels for the Newsela, Weebit, and Choosito corpora is shown in Table 26. As we can see, the grade distribution of Newsela articles is relatively inconsistent; there are over 1,000 articles at grade levels 4 and 12, but less than 50 at levels 2, 10, and 11.

With these corpora, we run six additional complexity prediction experiments: train and test on Choosito articles (in-domain), train on Newsela and test on Choosito (cross-domain); train on Choosito and test on Newsela (cross-domain); train and test on Weebit articles (in-domain); train on Newsela and test on Weebit (cross-domain); and train on Weebit and test on Newsela (cross-domain). The resulting confusion matrices are shown in Figure 11. As we can see, our models were again able to achieve consistently high performance in the in-domain experiments, but in the cross-domain settings the performance sharply decreases. A qualitative exploration of the data reveals that Weebit and Choosito are composed of noisier web articles which often contain more incomplete sentences and pictures captions. Hence, it seems that a significant change in text style has a significant impact on a model's ability to predict complexity. This is an important limitation to consider if our goal is to develop a generally-applicable complexity prediction model.

6.5. Re-ranking Retrieved Documents by Complexity

We use our in-domain complexity prediction model trained on Newsela (Section 6.4) to re-rank the documents in our initial retrieval experiments (Section 6.2). Our goal is to boost the documents that are predicted to be at the desired level in the ranking. In this experiment, we rank articles using the KAAD (Keep All Aligned Documents) configuration, which makes the task of finding the aligned document at the correct level more difficult.

During re-ranking, we first boost the rank of documents at the desired level (e.g., L0), followed by documents at the next closest level (L1), etc. We apply this re-ranking mechanism to both of our retrieval experiments, i.e. using only Newsela documents and augmenting the corpus with NYT articles.

6.5.1. Results

The results obtained after applying the re-ranking mechanism to the retrieval experiment restricted to Newsela articles are shown in Table 27. The results obtained in the setting where NYT articles are included are shown in Table 28. Prior to re-ranking, the L3 article is almost always ranked highest by both BERT and SBERT; this makes sense, as aligned documents at levels L3 and L4 generally are most similar. Interestingly, in Table 27, we see that SBERT ranks the L2 document as most similar to its corresponding L4 document 18% of the time (Top-1 Accuracy), while BERT does so a negligible 0.6% of the time. This again supports our previous finding that SBERT better captures semantic similarity across complexity levels.

We see that re-ranking significantly increases performance when retrieving L2, L1, and L0

Embedding Method	**Re-Ranked**	**Target Aligned Document CL (Newsela only)**							
		L3		**L2**		**L1**		**L0**	
		MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc
BERT	No	**0.995**	**0.993**	0.502	0.006	0.287	0	0.121	0
	Yes	0.784	0.782	**0.714**	0.645	**0.691**	0.639	0.616	0.521
SBERT	No	0.859	0.753	0.543	0.182	0.353	0.041	0.235	0.016
	Yes	0.776	0.767	**0.714**	0.654	**0.693**	**0.644**	**0.731**	**0.629**

Table 27: Ranking results for retrieving aligned Newsela documents at four complexity levels (CL), using the KAAD approach. For each embedding method, we report results with and without re-ranking.

Embedding Method	**Re-Ranked**	**Target Aligned Document CL (Newsela + NYT Distractors)**							
		L3		**L2**		**L1**		**L0**	
		MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc	MRR	Top1Acc
BERT	No	**0.993**	**0.989**	0.488	0.006	0.203	0	0.054	0
	Yes	0.784	0.782	**0.713**	**0.645**	**0.688**	**0.634**	0.599	0.512
SBERT	No	0.814	0.711	0.480	0.168	0.262	0.035	0.136	0.011
	Yes	0.770	0.760	0.701	0.638	0.675	0.622	**0.691**	**0.593**

Table 28: Ranking results for retrieving aligned Newsela documents at four complexity levels (CL), including the additional NYT distractor articles. We again use the KAAD approach, and report results with and without re-ranking.

documents. After incorporating re-ranking, SBERT no longer outperforms BERT for L2 and L1 documents, though it is still more effectively in identifying L0 documents. This demonstrates that a two-step process of embedding-based retrieval and re-ranking based on predicted complexity, can be effectively used to retrieve documents at any complexity level. On the other hand, re-ranking does not help when retrieving L3 articles. This is because when an article's complexity level is incorrectly predicted, relying on that prediction for re-ranking will lead to that article being pushed down in the list. However, since embedding-based methods are already quite effective at retrieving related documents at similar complexity levels, re-ranking is unnecessary in this case.

6.6. Summary

In this chapter, we present a reformulation of text simplification as a retrieval task. Our overall goal is to take in a document D and, after identifying concepts that are critical for understanding D, retrieve a document related to each concept that is more appropriate for a user. To this end, the first section of this chapter focuses on critical concept identification. We describe in detail various statistical approaches commonly used for the related task of keyphrase extraction, and propose several BERT-based approaches that utilize embedding similarity to rank concepts. We also present two baseline approaches that leverage Wikipedia categories. To test these methods, we create a high-quality concept identification evaluation corpus using trained annotators. We surprisingly find that both the statistical and Wikipedia-based baselines outperform our BERT-based approaches. This is likely because comparing a concept embedding is difficult to compare to a title or document embedding, which both contain many non-concepts.

In the second part of this chapter, we focus on the task of retrieving related documents at varying complexity levels. We show that embeddings generated from contextualized language models are able to effectively rank highly aligned documents from similar complexity levels, this is not the case when aligned documents are of more disparate complexities. To circumvent this issue, we propose a secondary ranking mechanism that predicts each document's complexity, and then leverages this prediction to re-rank the documents. With this, we are able to effectively retrieve related documents at any desired complexity level. This holds true even in a more challenging setting, where we have over one million distractor documents. While these results are indeed promising, we also discuss the fact that complexity can be domain-specific. We present cross-domain experiments which show the difficulty of fine-tuned complexity prediction models to maintain performance when transferring to a new domain.

CHAPTER 7 : Conclusion

The ability to automate the process of simplifying complex text is essential for children learning things for the first time, people with disabilities, second-language learners, and adults trying to shift to a new field in their career. Traditionally, the field of text simplification has addressed this as a generation task where given a complex text, a model needs to produce a simpler version that preserves its meaning. This is clearly a useful problem to solve, and also allows researchers to apply standard machine-translation-style generation models directly to this task. However, this framework does have inherent limitations; complexity is not a binary notion, and some phrases cannot just be replaced with simpler substitutes. This thesis has had three goals: to propose extensions to current simplification systems, while also identifying where these improvements still fall short; to systematically explore in detail the potential benefits of generating diverse candidates from a conditional language model; and, given the constraints of current simplification models, to propose a retrieval-based reformulation of the text simplification task to take steps towards a system that can be practically useful.

7.1. Summary of Contributions

The first part of this thesis focused on improving current lexical- and sentence-level text simplification systems. In Chapter 3, we focused on lexical simplification, which we broke down into two sub-tasks: identifying words that are difficult to understand in a text; and replacing them with simpler substitutes that make sense in context. For complex word identification, we train a Support Vector Machine classifier using word-based and context-based features, and evaluated its performance on our manually annotated dataset. We found that word-based features such as word length, frequency, and number of syllables already resulted in a competitive baseline, and while initial context-based features were not as useful, leveraging context via fine-tuning BERT resulted in further improvements. For the lexical simplification step, our proposed method revolved around extracting candidate substitutes from a large-scale simplification resource (Pavlick and Callison-Burch, 2016), and ranking them based on an embedding-based substitution model that takes into account the surrounding context (Melamud et al., 2015). We evaluated this approach on an evaluation set composed of aligned complex and simple sentence words, and found that it outperforms baselines that leverage other substitution resources or ranking mechanisms that do not rely on context

While in Chapter 3 we focus on a single text simplification sub-task, in Chapter 4 we attempt to holistically simplify entire sentences, leveraging a Sequence-to-Sequence (Seq2Seq) model to perform all simplification operations simultaneously. Previous work accurately noted that applying a generic Seq2Seq model to text simplification results in outputs that only slightly differed from the original sentence. Unlike previous Seq2Seq approaches, we propose ways to encourage the model to make operations that do not involve copying from the original during training, inference, and post-inference time, resulting in outputs that are shorter and simpler than in previous work. This does come at the cost of meaning preservation, which decreases human-annotated adequacy scores compared with the previous state-of-the-art of Zhang and Lapata (2017), and we explore this in an additional experiment that varies our choice of output based on word length.

In the second section of Chapter 4, we explore a novel quality estimation metric to more accurately measure the perform of sentence simplification systems. We propose a quality estimation model that is based on fine-tuning a BERT-based model (Devlin et al., 2019; Xenouleas et al., 2019) to simultaneously predict the fluency, adequacy, and relative complexity of an generated sentence simplification. We show that this results in a stronger correlation than previous metrics across all evaluation dimensions, and that training a single model to predict all three dimensions outperforms training three separate models.

In Chapter 4, our post-training extensions significantly improve the results, especially the mechanisms proposed for generating and re-ranking diverse candidate output sentences. In Chapter 5, we further explore the benefits of decoding diverse candidates from conditional language models, as this can have applications in different NLP generation tasks. We systematically compare a variety of diverse approaches that rely on beam-search and random sampling, on the tasks of conversational dialogue systems and image captioning. We find that outputs from diverse beam search methods are not actually that diverse, while outputs from random sampling methods contain significantly more diversity, but at the cost of quality. We explore this quality/diversity trade-off further, and find that over-sampling and selecting candidates based on post-decoding clustering or on model perplexity results in a reasonable balance between quality and diversity.

In Chapters 3 and 4, we focused on established tasks in the simplification field. However, despite making improvements over previous approaches in both cases, closer analyses revealed that our methods still have a long way to go before we are able to generate a coherent and simpler version of an entire document. This is not a surprising conclusion, as even subsequent language models trained on huge amounts of text have issues with long-term dependencies (Ippolito et al., 2020). Thus, in Chapter 6 we propose to reformulate text simplification as a retrieval task. Given a document D, our goals in this setting are to identify the concepts that are critical for understanding D, to retrieve a set of documents related to each concept, and to re-rank these in order to find related documents written at a more appropriate complexity level for a user.

The first section of Chapter 6 focuses on critical concept identification. We create a custom evaluation corpus, using trained annotators who were asked to identify concepts within a set of Computer Science web articles. We then use this dataset to evaluate the performance of unsupervised statistical approaches and several BERT-based methods. Interestingly, we find that statistical baselines outperform unsupervised methods relying on pre-trained language models. The second section of Chapter 6 focuses on the task of retrieving aligned documents at different complexity levels. Our methodology leverages an initial re-ranking mechanism that predicts document complexity which filters out those that are not at the desired level, before ranking the documents based on their embedding-based similarity to the original document. We show that an embedding-based approach can give good results when the desired level is close to that of the original complex document, but our re-ranking step is critical in retrieving aligned articles written at significantly different complexity levels. We conduct experiments in settings of varying difficulty, involving a diverse number of distractor documents, and show that our methodology successfully retrieves related documents at the desired complexity level even in a challenging scenario with over one million candidate documents.

7.2. Discussion and Future Work

There are two key conclusions of this thesis. The first, from Chapters 4 and 5, is that diverse decoding is helpful in a variety of NLP generation tasks, but how this is implemented matters and depends on the underlying model quality. If the model used is generally low quality, then attempting to generate diverse candidates will likely result in outputs that are mostly unintelligible. On the other hand, if we assume that the model produces high-quality output, then we can take advantage of the decoding techniques that encourage more diversity, mainly constrained random sampling. In addition, we showed that oversampling prior to candidate selection allows a better balance between quality and diversity.

The second conclusion is that while it is useful to continue improving upon the established task of sentence simplification, there may be a fundamental disconnect between this task and what is actually needed in order to help someone read a difficult article, or more generally, learn about an unfamiliar topic. Thus, in Chapter 6 this thesis proposes a retrieval-based reformulation of text simplification, showing how we can leverage both lighter weight and large-scale pre-trained models to identify concepts critical for understanding the content of a document, as well as for retrieving similar documents at different complexity levels. In order to continue to move the simplification field forward, it seems important to take a step back and think about the practical problems our models are actually trying to address.

This thesis leaves open several questions, which reflect the limitations of this work and open up opportunities for future research. First, more analysis should be done on the task-specific benefits and uses of generated diverse candidates. In the field of text simplification, there have been several recent works focusing on how to control the generated output, based on different priming tokens or sentence structures (Mallinson and Lapata, 2019a; Martin et al., 2020). In a similar vein, generating diverse high-quality simplifications could allow us to then select which simplification is appropriate for a specific situation.

Another question is how to actually represent a document's overall complexity. Based on our experiments described in Section 6.4, it is clear that we are able to accurately predict a document's complexity when training and testing on documents within a single corpus, but these models are relatively brittle when moving to another domain. To address this limitation, it would be interesting to consider more robust transfer learning options. Fine-tuning BERT is a type of transfer learning, but it still requires several hundred task-specific examples. In contrast, future work could consider leveraging models such as GPT-3 (Brown et al., 2020), which have shown to perform surprisingly well on few-shot and even zero-shot learning tasks, due to their vast increase in size over previous language models. This idea will naturally run into the problem of how to efficiently leverage models of this size in a practical setting, and also how to even train them in the first place. To begin to address this, future work can take inspiration from recent methods focusing on further optimizing the training and prediction of transformer-based models (Kitaev et al., 2020).

A natural extension of the retrieval experiments in Chapter 6 is to consider creating complexity-agnostic embedding representations, or representations that set aside specific dimensions for complexity, instead leveraging separate complexity and similarity ranking mechanisms. In this way, we would be able to fairly compare the similarity of any pair of texts, regardless of their complexity. One idea for doing this is to train a Generative Adversarial Network, or GAN; there has been recent work that creates two separate embedding representations of a text, one for the meaning of the text and one representing its style (Romanov et al., 2019). One potential issue with this approach is that it is often very challenging to separate these two components. In Section 6.1, we discuss a similar idea, that complex words and phrases are often vital to understanding the overall meaning of a text.

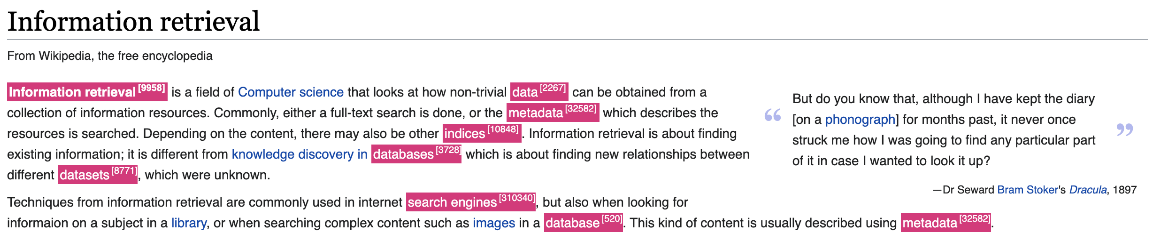
Finally, apart from gaining a better understanding of complexity in general, we also need to take into account the fact that different people approach the same text with different amounts of background knowledge. This significantly impacts the perceived complexity of that text. One way we can address this aspect is to have a personalized model of what people already know, based on what they have read in the past. However, even this is likely incomplete, because people learn about topics from many different media. This is clearly a difficult problem to solve, but a goal of the text simplification field should be to create models with a deeper understanding of this notion of user-dependent complexity. With this information, future work will be able to build powerful end-to-end resources to help anyone easily learn about a new topic, regardless of their background.

APPENDIX

A.1. Concept Identification Annotation Guidelines

In this section, we discuss the guidelines that we created for annotators to identify concepts critical to understanding a document.

- **Label nouns and compound nouns that express a concept critical relevant for understanding.** In some cases, only the head of the noun phrase, i.e. the noun itself, is the concept, while in other cases, a larger part of the noun phrase should be labeled as a concept.
  - **Example**: Using the classic <u>algorithms</u> of <u>machine learning</u>, text is considered as a sequence of keywords.
  - In this example, we label *algorithms* as a concept, but not *classic algorithms* or *the classic algorithms.* On the other hand, the phrase *machine learning* is a multi-word expression that represents a technical concept, so we must annotate the entire noun phrase in this case.
- **Only label noun phrases as concepts.** For our annotations, please do not just annotate an adjective or adverb. Instead, you should include it as part of a noun phrase.
  - **Example**: A three-dimensional vortex presents identical <u>perpendicular lines</u> at any given time.
- **Label common nouns that are used in specific technical contexts.** In certain domains, normally common words acquire a technical meaning that warrant them being labeled as a concept.
  - **Example**: Machine learning is concerned with minimizing the <u>loss</u> on unseen examples...a representative book was Nilsoon's *Learning Machines.*
  - In this example, *loss* is a concept, as even though it is a common word, it is used in an uncommon, machine learning-specific context. On the other hand, *book* is



Information retrieval

From Wikipedia, the free encyclopedia

Information retrieval[9958] is a field of Computer science that looks at how non-trivial data[2267] can be obtained from a collection of information resources. Commonly, either a full-text search is done, or the metadata[32582] which describes the resources is searched. Depending on the content, there may also be other indices[10848]. Information retrieval is about finding existing information; it is different from knowledge discovery in databases[3728] which is about finding new relationships between different datasets[8771], which were unknown.

Techniques from information retrieval are commonly used in internet search engines[310340], but also when looking for informaion on a subject in a library, or when searching complex content such as images in a database[520]. This kind of content is usually described using metadata[32582].

“But do you know that, although I have kept the diary [on a phonograph] for months past, it never once struck me how I was going to find any particular part of it in case I wanted to look it up?”

—Dr Seward Bram Stoker's *Dracula*, 1897

Figure 12: An example of concept annotations, collected using our annotation tool.

not a concept in this context.

- **When a concept appears along with a prepositional phrase, label the full phrase.** Some nouns are followed by prepositional phrases describing the noun. For these, please label the entire phrase, including the preposition.
  - **Example**: Each section is perpendicular to its <u>axis of rotation</u>...This area is usually called the <u>core of the reverberator</u>.
- **Annotate all distinct versions of concepts.** There are some expressions which, while they may overlap in word usage, represent different concepts.
  - **Example**: <u>Regression analysis</u> encompasses a variety of statistical methods...the most common type of <u>regression</u> is <u>linear regression</u>.
  - Make sure to annotate both a shorter and a longer form if both are concepts that appear in different places. In this example, *regression* is labeled as a concept, even though it is part of the more specific *linear regression* concept term found elsewhere.

A.2. Simplification as Retrieval Error Analysis Examples

In Chapter 6, we show a partial example of aligned Newsela articles that significantly alter content. Here, we show the full text of several such (D_{i4}, D_{i0}) document pairs, where D_{i0} was ranked outside the top 500 most similar documents. We show two document pairs missed by our BERT-based retrieval method, and one document pair missed by our SBERT-based retrieval method.

Source L4 Document
Facebook Chief Executive Mark Zuckerberg announced Tuesday that he plans to eventually donate 99 percent of the Facebook stock owned by him and his wife, Priscilla Chan, shares that are worth about $45 billion today. That amount would make it one of the largest philanthropic commitments ever. Zuckerberg said in an open letter that the birth last week of their first child, a daughter named Max, was the motivation to dedicate their considerable wealth to charitable causes now. They wrote that they did not want to wait to "advance human potential and promote equality for all children." "I will continue to serve as Facebook's CEO for many, many years to come," Zuckerberg wrote, "but these issues are too important to wait until you or we are older to begin this work." The money will be channeled into the Chan Zuckerberg Initiative, a newly formed group that initially will focus on education and health. It was also clear from Zuckerberg's letter that he and his wife had learned lessons from earlier philanthropic attempts and will move in a new direction. Zuckerberg's announcement stands out because of his relative youth – he's just 31 – and the gift's massive size. The donation also comes at a time when the world's wealthiest business leaders seem to be challenging one another to give away their fortunes before they die. In 2010, Bill Gates and Warren Buffett publicly launched the Giving Pledge to encourage billionaires to donate the bulk of their wealth to charity. Gates, the former Microsoft leader, has already donated $31.5 billion, mostly to the Bill and Melinda Gates Foundation. Buffett has donated more than $22 billion of his Berkshire Hathaway stock and plans to give away nearly his entire fortune. More than 130 billionaires worldwide have joined them, including Judy Faulkner, founder of the electronic health-records company Epic, who reportedly said she plans to give away 99 percent of her money. Also this year, Saudi Arabia's Prince Alwaleed bin Talal, one of the richest men in the world, said the pledge inspired him to eventually give away his entire fortune, more than $30 billion. Zuckerberg quietly committed weeks ago to this pledge, too, although his Nov. 9 Giving Pledge letter didn't reveal the scale of his intentions. A federal filing shows the Facebook co-founder will start by giving away up to $1 billion a year in Facebook stock for the first three years. Such tax-efficient arrangements are not unusual for gifts to private foundations. "He clearly wants to be perceived as a leader of his generation in the same way as Buffett and Gates are theirs," said Richard Marker of the New York advisory firm Wise Philanthropy. The couple revealed their plans in a lengthy "letter to our daughter" published on Facebook, the social network that Zuckerberg co-founded while a student at Harvard University and that today has 1.5 billion monthly active users worldwide. Their daughter's full name is Maxima Chan Zuckerberg, and she was born just before Thanksgiving, weighing 7 pounds, 8 ounces, according to a Facebook spokeswoman. In the open letter, Zuckerberg and Chan talked about the potential that technology offers to re-engineer the way children learn. "Our generation grew up in classrooms where we all learned the same things at the same pace regardless of our interests or needs," they wrote. Zuckerberg and Chan have given $15 million to AltSchool, a for-profit corporation founded by a former Google executive that works to create a network of schools that use technology to personalize instruction. The couple also has given $120 million to traditional public schools and public charter schools in the San Francisco area. But Zuckerberg's first foray into education philanthropy was in itself a learning experience. In 2010, he gave $100 million to remake public schools in Newark, New Jersey, with mixed results. Critics of that effort said the plan faltered in part because it was a top-down approach that called for wholesale changes in the city's public schools with plans crafted by outsiders, not community members. In their letter to Max, Zuckerberg and Chan talked about learning from mistakes. "Your mother and I have both taught students and we've seen what it takes to make this work. It will take working with the strongest leaders in education to help schools around the world adopt personalized learning. It will take engaging with communities, which is why we're starting in our San Francisco Bay Area community." Zuckerberg and Chan, a pediatrician and onetime teacher, plan to open a private, tuition-free school for low-income children in East Palo Alto, California, that will integrate health care and support for families with classroom learning. Although the letter to Max was obviously intended for a wide audience, revealing their ambition to fund billions of dollars in new philanthropic works, the couple ended the note in a personal, quiet fashion. It was signed simply: "Love, Mom and Dad."

Missed Aligned L0 Document
Mark Zuckerberg runs Facebook. He helped start the company when he was a in college. Facebook is used by billions of people around the world. It has made Mark a great deal of money. He is very, very rich. Mark had some big news his week. He said he and his wife will give away much of their money. His wife's name is Priscilla Chan. She is a doctor. The two promised to give away around $45 billion. The money will be used to help make schools better. ## News Was Delivered On Facebook Mark said this on Facebook. It was in a letter to his first child. Her name is Maxima Chan Zuckerberg. Maxima was born in November. Her mother and father call her Max for short. Mark said it was time to start giving to others. He and his wife want to help all children be the best they can be. Mark and Priscilla said computers can make schools better. They can help kids learn. There are many new ideas on how to use computers and other technology. ## Learning That Fits Each Student The two said they "grew up in classrooms where we all learned the same things." Some students learned more quickly than others. Students were interested in different things. Everyone learned the same things in the same way. Teaching that way can be a problem, they said. Some students do not do as well as they can. Others are bored. ## Better Teaching With Technology Technology can make teaching better, Mark and Priscilla said. It makes teaching different for each student. All kids learn in the way that is best for them. The money Mark and Priscilla are giving away will help schools use new technology. It will help them to do better in school. Mark and Priscilla are also opening a free school for poor children. The school will be in California.

Table 29: Example 1 of an aligned document pair missed by our BERT-based model, titled *zuckerberg-donation*.

Source L4 Document

ANCHORAGE - Anna Oxereok grew up eating walrus in the western Alaska village of Wales. Today it's such a rare treat she can't bring herself to part with the plastic gallon bag of meat in her freezer.
“I have to save it for something special,” she says.
Her brother caught two animals this spring and shared the meat and fat, but it didn't go very far in the village of 150. She is thankful for what she got, though. It's become increasingly difficult to land a walrus.
Other remote communities at the edge of the Bering Sea also are seeing a steep decline in walrus harvested the past several years.
Walrus, described by some as having a taste between veal and beef, is highly prized by Alaska Natives as a subsistence food to store for winter, with the adult male animals averaging 2,700 pounds. The sale of carved ivory from the tusks, legal only for Alaska Natives, also brings in supplemental income to communities with high unemployment rates.
Hunters and scientists say walrus migration patterns are veering from historical hunting grounds as temperatures rise and the ocean ice used by the animals to dive and rest recedes farther north. Village elders also tell biologists the wind is blowing in new directions. In 2013, a late-season icepack clustered around St. Lawrence Island, blocking hunters from the sea.
“I think one of the biggest issues is that things have gotten so variable. It's hard to really predict what's going to happen,” said Jim MacCracken, Alaska walrus program supervisor for the U.S. Fish and Wildlife Service.
Iver Campbell and other Yup'ik Eskimo hunters from two St. Lawrence Island communities harvested more than 1,100 walrus in 2003. But a decade later, hunters managed to take only 555 - a fraction of the ideal of one walrus per resident, per year. Things still are not looking any better for the 1,430 residents of the villages of Gambell and Savoonga. The recent spring take was 233 walrus, according to preliminary Fish and Wildlife figures.
The shore ice once served to block the wind for hunters but that's no longer the case, said Campbell, who has lived all 64 years in Gambell, population 713.
“The ice goes out real fast, melts real fast,” he said. “We don't have anything to counter the wind and the rough water.”
Science backs that observation. According to the Office of Naval Research, the past eight years have had the eight lowest amounts of summer sea ice on record.
Far from the state's limited road system, costly store-bought food is not an affordable solution. At village stores, pantry staples quickly add up - nearly $7 for a dozen eggs, $15 for a gallon of milk and $6.25 for a loaf of basic white bread. People rely on the region's resources for up to 80 percent of their diets.
Their hunting practices are closely monitored by federal authorities to ensure the animals that are killed are not going to waste. Generally, such hunts do not cause a public outcry in Alaska.
In these communities, a subsistence lifestyle is a necessity. In fact, the low harvest this year recently prompted a donation of 10,000 pounds of frozen halibut to four affected villages.
“A decline in the subsistence harvest really creates an economic disaster that threatens the health and welfare of the people in the communities,” said Vera Metcalf, director of the Eskimo Walrus Commission. “So we are concerned about the impacts of climate change and the ability for our hunters to harvest marine mammals.”
Some Native communities can search for other animals, like domestic reindeer or caribou. But opportunities aren't as bountiful for Diomede on the western coast of Little Diomede Island, only a few miles from Russia. The community of 120 harvested one walrus in 2014, prompting city and Native leaders to seek assistance from the state.
This year, 10 walrus were harvested, according to Diomede hunter Robert Soolook. There's no shortage of walrus, he said, but they are migrating sooner. No one has initiated any long-range planning to address the shift, but Soolook believes hunters eventually will need to change their practices, even going out earlier.
“Now that we've seen this, we have to start adapting,” he said.
No federal assistance is available, and state aid is minimal, at best. State Sen. Donny Olson, a Democrat from Golovin, said he might introduce legislation to allow failed subsistence hunts to qualify for state disaster funds.
Moving from her ancestral lands is not an option, according to Oxereok, an Inupiat Eskimo. Relocating would mean displacing everything she knows.
“It's not that simple because your roots are here,” she said.

Missed Aligned L0 Document

ANCHORAGE, Alaska - There are people who have been living in Alaska for many, many years. They were living there long before the United States began. They are called Native Alaskans.
Native Alaskans eat walrus. Walrus are large sea animals. Only Native Alaskans are allowed to hunt walrus. Eating walrus is part of their way of life.
These days, walrus are hard to find. There are not as many in Alaska anymore. Global warming has made many walrus swim away. The ice they like is melting. Global warming means that our planet is getting hotter.
A Special Piece Of Meat
If there are no walrus to eat, Native Alaskans will not have enough food. Some people are going hungry.
Anna Oxereok is a Native Alaskan. She grew up eating walrus in her village. Today, she does not eat much walrus. Anna has one piece of walrus in her freezer. She is waiting to eat it.
“I have to save it for something special,” she says.
Her brother caught two walrus this spring. He shared them with their village. Two walrus for 150 people was not very much. Anna was glad she got some.
Walrus Look For Ice
Walrus are moving around more. Global warming has forced them to. In Alaska, there is ice in the sea. Walrus like to rest on the ice. Now it is getting warmer in Alaska. More ice has turned to water. Some walrus have moved. They went to where there is still ice. Now there are fewer walrus for Native Alaskans to hunt.
Without walrus to eat, there has been less meat. Native Alaskans do not buy much food in stores. They do not have enough money. They must find their own food. Finding enough food has been hard without walrus.
Anna is not thinking about moving. Moving would mean leaving everything she knows. Anna will stay. She hopes that enough walrus stay, too.

Table 30: Example 2 of an aligned document pair missed by our BERT-based model, titled *walrus-alaska.*

Source L4 Document

WASHINGTON - As proof that he can successfully and humanely deport the estimated 11 million people living in the country illegally, Republican presidential contender Donald Trump often touts the efforts of the Eisenhower administration in the 1950s. He did so again during the Republican debate on Nov. 10, saying “you don’t get nicer, you don’t get friendlier” than President Dwight D. Eisenhower. “They moved 1.5 million out,” the billionaire real estate mogul said. “We have no choice. We have no choice.”
But the program to which Trump refers, known as “Operation Wetback,” was a complicated undertaking largely viewed by historians as a dark moment in America’s past. Also lost in Trump’s telling is that it coincided with a guest worker program that provided legal status to hundreds of thousands of largely Mexican farm workers.
Trump declined to refer to the program by name on Nov. 11 in an interview on The O’Reilly Factor. “I don’t like the term at all,” he said.
But he nonetheless defended what host Bill O’Reilly described as brutal treatment of those who were deported.
“I’ve heard it both ways. I’ve heard good reports, I’ve heard bad reports,” Trump said. “We would do it in a very humane way.”
“He’s only got part of the story,” said Mae Ngai, a professor of history at Columbia University.
The operation was named after a term for Mexicans who crossed the Rio Grande that is now viewed a racial slur. The 1954 initiative was aimed at apprehending and deporting agricultural workers who had crossed the border illegally looking for work.
According to a summary of the project from the Texas State Historical Association, the United States Border Patrol “aided by municipal, county, state, and federal authorities, as well as the military, began a quasi-military operation of search and seizure of all unauthorized immigrants.”
The project, Ngai said, began with 750 immigration officers and border control agents, who used jeeps, trucks, buses and airplanes to apprehend migrants nationwide, including in Los Angeles, San Francisco and Chicago. They apprehended 3,000 people a day and 170,000 during its first three months.
In an interview on Nov. 11 on MSNBC’s “Morning Joe,” Trump indicated he would take a similar approach. ”You’re going to have a deportation force, and you’re going to do it humanely,” he said.
Critics of the program say the conditions for those the agents apprehended were anything but humane. Many of the apprehended migrants were transported in crowded buses and dumped on the other side of the border in a manner that some at the time equated with the treatment of livestock.
In one incident, Ngai said, 88 apprehended Mexicans died of sunstroke after being subjected to 112-degree heat. The number would have been higher had the Red Cross not intervened.
Some of those apprehended were sent deep into the interior of Mexico to prevent re-entry by train or cargo ship, where conditions drew the attention of federal regulators.
One congressional investigation likened a transport ship that was the site of a riot to an “18th century slave ship” and a “penal hell ship.”
Trump touted the approach as a virtue of the Eisenhower-era program in the Nov. 10 debate.
“Moved a 1.5 million illegal immigrants out of this country, moved them just beyond the border. They came back. Moved them again beyond the border, they came back. Didn’t like it,” Trump said. ”Moved them way south. They never came back.”
Trump also leaves out of his advocacy for the Eisenhower-era approach the fact the program was developed to complement a guest-worker program that began in the 1940s and was aimed at allowing Mexican farmworkers to enter the country and work in the United States legally.
Hundreds of thousands of farm workers did so, and the deportation effort was conceived as a way to pressure employers into using the guest worker program.
“It was like a carrot and a stick,” Ngai said.
While Trump has put the number of deportations at 1.5 million, most accounts suggest the numbers are far fewer, because they included those who chose to leave the country voluntarily as well as people who returned after being deported and were deported again.
Trump has yet to lay out precisely how he would track down those living in the country illegally, or how he would determine who are “the good ones” that he would allow to return. Both John Kasich, Ohio’s governor, and Jeb Bush, the former governor of Florida, rejected Trump’s plan as unrealistic and cruel in the Republican presidential candidates’ debate on Tuesday night, Nov. 10.
“To send them back, 500,000 a month, is just not, not possible,” Bush said. “And it’s not embracing American values. And it would tear communities apart. And it would send a signal that we’re not the kind of country that I know America is.”

Missed Aligned L0 Document

WASHINGTON, D.C. - Donald Trump is running for president.
The race for president is in two parts. The first part is known as the primaries. Republican candidates run against other Republicans in the Republican primary. Democrats run against Democrats in the Democratic primary.
Then, the winner of the Republican primary runs against the winner of the Democratic primary. The second race happens in 2016. It is known as the election. The people picked to run in that race are called the nominees. The winner of the election becomes president.
On Nov. 10, Republican candidates got together for a debate. They all said what they would do if elected president. They gave speeches on TV. Trump is a Republican. He said that millions of people are living in the country against the law. He would make them all leave.
Trump Likes One-Way Trips
Trump praised a program from the 1950s. It forced farmworkers to leave the United States. Most of them were from Mexico. They crossed the border without permission. They were looking for work.
Trump likes the program. He said it was nice and friendly. He called it proof that he could send farmworkers back to their own countries in a kind way.
Historians Oppose Trump’s View
Historians do not agree with Trump. Most of them call the 1950s program an ugly moment in America’s past.
The program began with 750 government officers. They used jeeps, trucks, buses and airplanes to find the farm workers. The officers caught 3,000 people a day. They caught about 170,000 during the program’s first three months.
Many people say the government treated the farmworkers badly. The government put them on crowded buses. Then they were dumped on the Mexican side of the border.
At one point, 88 Mexicans died, said Mae Ngai, a history professor. The Mexican workers were left outside in very hot weather.
GOP Rivals Criticize Trump’s Plan
Two Republican candidates spoke out against Trump’s plan. One of the candidates was Ohio Governor John Kasich. The other one was Jeb Bush. He used to be the governor of Florida. Both of them said Trump’s plan was cruel. They also said it would be hard to carry out.
Bush said Trump’s plan is against American values. He said that America is a better country than that.

Table 31: Example of an aligned document pair missed by our SBERT-based model, titled *trump-immigration*.

BIBLIOGRAPHY

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*, pages 49–54, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-3009. URL `https://www.aclweb.org/anthology/D19-3009`.

P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, 2016.

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.

M. Apidianaki. Vector-space models for PPDB paraphrase ranking in context. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2028–2034, Austin, Texas, 2016.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*, 2015.

A. Baheti, A. Ritter, J. Li, and B. Dolan. Generating more interesting responses in neural conversation models with distributional constraints. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, 2018.

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219914. URL `https://www.aclweb.org/anthology/P05-1074`.

E. Bernhardsson. *Annoy: Approximate Nearest Neighbors in C++/Python*, 2018. URL `https://pypi.org/project/annoy/`. Python package version 1.13.0.

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

O. Bojar, C. Federmann, B. Haddow, P. Koehn, M. Post, and L. Specia. Ten years of wmt evaluation campaigns: Lessons learnt. 2016.

O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck,

P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, and M. Turchi. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4717. URL `https://www.aclweb.org/anthology/W17-4717`.

T. Brants and A. Franz. Web 1t 5-gram version 1. In *LDC2006T13*, Philadelphia, Pennsylvania, 2006. Linguistic Data Consortium.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. 2020.

C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluation the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006. URL `https://www.aclweb.org/anthology/E06-1032`.

R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509: 257–289, 2020.

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999. URL `https://www.aclweb.org/anthology/E99-1042`.

R. Chandrasekar and S. Bangalore. Automatic induction of rules for text simplification. *Knowl.-Based Syst.*, 10:183–190, 1997.

T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *CoRR*, abs/1512.01274, 2015.

K. Cho. Noisy parallel approximate decoding for conditional recurrent language model. 2016.

S. Choudhary, P. Srivastava, L. H. Ungar, and J. Sedoc. Domain aware neural dialog system. volume abs/1708.00897, 2017.

E. Clark, A. S. Ross, C. Tan, Y. Ji, and N. A. Smith. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, pages 329–340, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-4945-1. doi: 10.1145/3172944.3172983. URL `http://doi.acm.org/10.1145/3172944.3172983`.

K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

A. Cocos and C. Callison-Burch. Clustering paraphrases by word sense. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1463–1472, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1172. URL `https://www.aclweb.org/anthology/N16-1172`.

A. Cocos, M. Apidianaki, and C. Callison-Burch. Word Sense Filtering Improves Embedding-Based Lexical Substitution. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 110–119, Valencia, Spain, April 2017.

W. Coster and D. Kauchak. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon, June 2011. Association for Computational Linguistics.

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *NIPS*, 2015.

C. Danescu-Niculescu-Mizil and L. Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W11-0609`.

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

S. Devlin and J. Tait. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173, 1998.

G. R. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Human Language Technology: Notebook Proceedings*, pages 128–132, 2002.

A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P18-1082`.

M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1606–1615, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1184. URL `https://www.aclweb.org/anthology/N15-1184`.

D. Feblowitz and D. Kauchak. Sentence simplification as tree transduction. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 1–10, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W13-2901`.

K. Filippova and M. Strube. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32, Salt Fork, Ohio, USA, 2008. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W08-1105`.

J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia, 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N13-1092`.

K. Gimpel, D. Batra, C. Dyer, and G. Shakhnarovich. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, 2013.

Y. Graham. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal, 2015. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D15-1013`.

J. Gu, K. Cho, and V. O. Li. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1210. URL `https://www.aclweb.org/anthology/D17-1210`.

S. Gupta and C. Manning. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1–9, Chiang Mai, Thailand, Nov. 2011. Asian Federation of Natural Language Processing. URL `https://www.aclweb.org/anthology/I11-1001`.

T. B. Hashimoto, H. Zhang, and P. Liang. Unifying human and statistical evaluation for natural language generation. *CoRR*, abs/1904.02792, 2019.

K. Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, UK, 2011.

J. Hewitt and C. D. Manning. A Structural Probe for Finding Syntax in Word Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL `https://www.aclweb.org/anthology/N19-1419`.

F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, and M. Post. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690, 2017.

A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL `https://www.aclweb.org/anthology/P18-1031`.

A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, 2003. URL `https://www.aclweb.org/anthology/W03-1028`.

D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1365. URL `https://www.aclweb.org/anthology/P19-1365`.

D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.164. URL `https://www.aclweb.org/anthology/2020.acl-main.164`.

C. Jiang, M. Maddela, W. Lan, Y. Zhong, and W. Xu. Neural CRF model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.709. URL `https://www.aclweb.org/anthology/2020.acl-main.709`.

D. Kauchak. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1537–1546, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the*

2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, 2014. Association for Computational Linguistics.

J. P. Kincaid, R. P. Fishburne, R. E. 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 ; research branch report 8-75. 1975.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

N. Kitaev and D. Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1249. URL `https://www.aclweb.org/anthology/P18-1249`.

N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL `https://doi.org/10.18653/v1/P17-4012`.

J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3337–3345. IEEE, 2017.

R. Kriz, E. Miltsakaki, M. Apidianaki, and C. Callison-Burch. Simplification using paraphrases and context-based lexical substitution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 207–217, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1019. URL `https://www.aclweb.org/anthology/N18-1019`.

R. Kriz, J. Sedoc, M. Apidianaki, C. Zheng, G. Kumar, E. Miltsakaki, and C. Callison-Burch. Complexity-weighted loss and diverse reranking for sentence simplification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3137–3147, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1317. URL `https://www.aclweb.org/anthology/N19-1317`.

R. Kriz, M. Apidianaki, and C. Callison-Burch. Simple-qe: Better automatic quality estimation for text simplification. *ArXiv*, 2020.

I. Kulikov, A. H. Miller, K. Cho, and J. Weston. Importance of a search strategy in neural dialogue modelling. 2018.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages 1188–1196. JMLR.org, 2014.

J. Li and D. Jurafsky. Mutual information and diverse decoding improve neural machine translation. *ArXiv*, abs/1601.00372, 2016.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, June 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL `https://www.aclweb.org/anthology/N16-1014`.

J. Li, W. Monroe, and D. Jurafsky. A Simple, Fast Diverse Decoding Algorithm for Neural Generation. *CoRR*, 2016b.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

R. Luo. An image captioning codebase in pytorch, 2017.

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL `https://www.aclweb.org/anthology/D15-1166`.

J. Mallinson and M. Lapata. Controllable sentence simplification: Employing syntactic and lexical constraints. *ArXiv*, abs/1910.04387, 2019a.

J. Mallinson and M. Lapata. Controllable sentence simplification: Employing syntactic and lexical constraints. *ArXiv*, abs/1910.04387, 2019b.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, 2014.

L. Martin, S. Humeau, P.-E. Mazaré, É. de La Clergerie, A. Bordes, and B. Sagot. Reference-less Quality Estimation of Text Simplification Systems. In *Proceedings of the 1st Workshop on Automatic Text Adaptation (ATA)*, pages 29–38, Tilburg, the Netherlands, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-7005. URL `https://www.aclweb.org/anthology/W18-7005`.

L. Martin, B. Sagot, É. de la Clergerie, and A. Bordes. Controllable sentence simplification. In *LREC*, 2020.

A. F. T. Martins, M. Junczys-Dowmunt, F. N. Kepler, R. Astudillo, C. Hokamp, and R. Grundkiewicz. Pushing the Limits of Translation Quality Estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218, Dec. 2017. doi: 10.1162/tacl_a_00056. URL `https://www.aclweb.org/anthology/Q17-1015`.

D. McCarthy and R. Navigli. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic, 2007.

O. Melamud, O. Levy, and I. Dagan. A Simple Word Embedding Model for Lexical Substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado, 2015.

R. Mihalcea and P. Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W04-3252`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, 2013b.

G. A. Miller. Wordnet: A lexical database for english. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992. URL `https://www.aclweb.org/anthology/H92-1116`.

G. A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38 (11):39–41, 1995.

R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL `https://www.aclweb.org/anthology/K16-1028`.

C. Napoles, B. Van Durme, and C. Callison-Burch. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97, Portland, Oregon, 2011. Association for Computational Linguistics.

C. Napoles, M. Gormley, and B. Van Durme. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montreal, Canada, 2012.

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)*, pages 435–445, Baltimore, Maryland, 2014. Association for Computational Linguistics.

P. Nathan. PyTextRank, a Python implementation of TextRank for phrase extraction and summarization of text documents. Derwen, 2016. doi: 10.5281/zenodo.4602393. URL `https://github.com/DerwenAI/pytextrank`.

R. Nelken and S. M. Shieber. Towards robust context-sensitive sentence alignment for monolingual corpora. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006. URL `https://www.aclweb.org/anthology/E06-1021`.

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)*, pages 85–91, Vancouver, Canada, 2017. Association for Computational Linguistics.

G. Paetzold and L. Specia. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California, 2016. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda. English Gigaword Fifth Edition LDC2011T07. DVD. *Philadelphia: Linguistic Data Consortium*, 2011.

A. Patel, A. Sands, C. Callison-Burch, and M. Apidianaki. Magnitude: A fast, efficient universal vector embedding utility package. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 120–126, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2021. URL `https://www.aclweb.org/anthology/D18-2021`.

E. Pavlick and C. Callison-Burch. Simple PPDB: A Paraphrase Database for Simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, Berlin, Germany, 2016.

E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, 2015.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://www.aclweb.org/anthology/N18-1202`.

M. Popel and O. Bojar. Training tips for the transformer model. *arxiv*, 2018.

A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.

L.-A. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL*, 2011.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL `https://www.aclweb.org/anthology/D19-1410`.

A. Romanov, A. Rumshisky, A. Rogers, and D. Donahue. Adversarial decomposition of text representation. In *Proceedings of NAACL 2019: Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. In *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd, 2010. ISBN 9780470689646.

H. Saggion. *Automatic Text Simplification.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24:513–523, 1988.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

M. Shardlow. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria, Aug. 2013a. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P13-3015`.

M. Shardlow. The CW corpus: A new resource for evaluating the identification of complex words. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77, Sofia, Bulgaria, Aug. 2013b. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W13-2908`.

P. Sharma and Y. Li. Self-supervised contextual keyword and keyphrase retrieval with self-labelling. 2019.

A. Siddharthan. A survey of research on text simplification. In *Special issue of International Journal of Applied Linguistics*, volume 165, 2014.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA, 2006.

L. Specia, S. K. Jauhar, and R. Mihalcea. Semeval-2012 task 1: English lexical simplification. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 347–355, Montréal, Canada, 2012.

L. Specia, F. Blain, V. Logacheva, R. Astudillo, and A. F. T. Martins. Findings of the WMT 2018 Shared Task on Quality Estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709, Belgium, Brussels, 2018. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W18-6451`.

S. Stajner, M. Popovic, and H. Bchara. Quality Estimation for Text Simplification. In *Proceedings of the LREC Workshop on Quality Assessment for Text Simplification*, pages 15–21, Portoroz, Slovenia, June 2016.

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, pages 738–744, Brussels, Belgium, 2018. Association for Computational Linguistics.

M. A. Sultan, S. Bethard, and T. Sumner. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230, 2014.

H. Sun and M. Zhou. Joint learning of a dual SMT system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P12-2008`.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Neural Information Processing Systems*, Montreal, Canada, 2014.

Y.-C. Tam, J. Ding, C. Niu, and J. Zhou. Cluster-based beam search for pointer-generator chatbot grounded by knowledge. In *Dialog System Technology Challenges 7 at AAAI 2019*, 2019.

S. R. Thomas and S. Anderson. Wordnet-based lexical simplification of a document. In J. Jancsary, editor, *Proceedings of KONVENS 2012*, pages 80–88. ÖGAI, 2012.

J. Tiedemann. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248, 2009.

T. Tom Kenter and M. de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, page 1411?1420. Association for Computing Machinery, 2015.

C.-T. Tsai, G. Kundu, and D. Roth. Concept-based analysis of scientific literature. In *CIKM*, 2013.

P. Turney. Learning to extract keyphrases from text. *Information Retrieval*, 2(4):303–336, 2000. URL `http://cogprints.org/1802/`.

S. Upadhyay, N. Gupta, and D. Roth. Joint multilingual supervision for cross-lingual entity linking. In *EMNLP*, 2018.

S. Vajjala and D. Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173, Montréal, Canada, June 2012. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W12-2019`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Neural Information Processing Systems*, Long Beach, CA, 2017.

A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. 2016.

O. Vinyals and Q. V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

T. Vu, B. Hu, T. Munkhdalai, and H. Yu. Sentence simplification with memory-augmented neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 79–85, 2018.

K. Woodsend and M. Lapata. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK., 2011. Association for Computational Linguistics.

S. Wubben, A. van den Bosch, and E. Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea, 2012. Association for Computational Linguistics.

S. Xenouleas, P. Malakasiotis, M. Apidianaki, and I. Androutsopoulos. SUM-QE: a BERT-based Summary Quality Estimation Model. 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)*, pages 6004–6010, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1618. URL `https://www.aclweb.org/anthology/D19-1618`.

J. Xu, X. Ren, J. Lin, and X. Sun. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, 2018.

W. Xu, C. Callison-Burch, and C. Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3(1): 283–297, 2015.

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*, 4(1):401–415, 2016.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. In *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

X. Zhang and M. Lapata. Sentence simplification with deep reinforcement learning. In *Pro-*

ceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 584–594. Association for Computational Linguistics, 2017.

Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, and B. Dolan. Generating informative and diverse conversational responses via adversarial information maximization. In *Advances in Neural Information Processing Systems*, pages 1815–1825, 2018.

S. Zhao, R. Meng, D. He, S. Andi, and P. Bambang. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 EMNLP Conference*, pages 3164–3173, Brussels, Belgium, 2018.

Z. Zhu, D. Bernhard, and I. Gurevych. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China, 2010.