Pomona College
Department of Computer Science

Crowdsourcing Text Simplification with Sentence Fusion

Max Schwarzer

April 29, 2018

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Computer Science

Professor David Kauchak, advisor

Abstract

In this paper, we present and evaluate a system for conducting text simplification using crowdsourcing, employing amateur human workers to simplify text at a sentence level. We also introduce a graph-based sentence fusion system, which we use to augment the output of the human workers. Finally, an SVM-based reranking system is presented, which enables the user to select the desired balance between simplification and meaning preservation, and a number of features are introduced and examined. We use the Newsela dataset [Xu et al., 2015] as a benchmark, and demonstrate consistent improvements at all simplification levels. Finally, we demonstrate that the inclusion of the sentence fusion system allows for significantly more simple output.

Acknowledgments

I would like to thank Professor David Kauchak, for his devotion and patience as my advisor. I would also like to thank my parents, brother, and partner for their support throughout my undergraduate career.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, I introduce a system for crowdsourcing text simplification. Up to today, research on text simplification has largely focused on automated systems, such as systems employing phrase-based machine translation [Coster and Kauchak, 2011], neural network machine translation models [Nisioi et al., 2017] and even deep reinforcement learning [Zhang and Lapata, 2017].

However, there has been a dearth of research into how *human* text simplification might be optimized. As automatic text simplification systems have yet to be widely adopted, and most simplification is still conducted exclusively by humans, this represents a significant gap in the literature. Indeed, as quality of output is very important in text simplification, due to the poor reading skills of much of the target audience, human-based manual text simplification will likely remain important [Siddharthan, 2014].

Therefore, we adopt a research direction which has previously been thoroughly explored in machine translation, crowdsourcing. Research has shown that users of Amazon Mechanical Turk are able to create high-quality translations from other languages into English [Callison-Burch, 2009; Zaidan and Callison-Burch, 2011]. However, only limited work has been conducted on directly crowdsourcing text simplification. Both Amancio and Specia [2014] and Lasecki et al. [2015] conducted analysis of simplifications gathered with amateur workers on Amazon Mechanical Turk, but neither constructed a complete pipeline for conducting text simplification automatically.

We do so. Moreover, we also introduce a sentence fusion system, modeled on the work of Filippova [2010], which we use to augment the simplifications produced by our amateur workers. We then employ a number of features, including a neural language model, to train an SVM-based

Original	The infrared data, for example, revealed that interstellar space is filled with diffuse polycyclic aromatic hydrocarbon gas.
Simplification	Infrared data revealed that space is filled with hydrocarbon gasses.
Simplification	Interstellar space is filled with diffuse polycyclic aromatic hydrocarbon gas.
Simplification	The heat sensor information showed that space has lots of stinky gas.
Simplification	The infrared data revealed interstellar space his filled with diffuse polycyclic aromatic hydrocarbon gas.
Fusion	Infrared data revealed interstellar space has lots of stinky gas.

Table 1.1: An example of crowdsourced simplifications, along with a fusion sentence produced by our system. Our reranked selected the fusion sentence. Note that the fusion sentence is qualitatively different from the human input sentences; this relationship will be examined in depth in later chapters.

ranker [Herbrich et al., 1999; Pedregosa et al., 2012] to select among our candidate simplifications. Using this ranker, we demonstrate that we are able to convincingly beat our benchmark of professional simplifications, Newsela [Xu et al., 2015]. We also show that our sentence fusion system very significantly increases the ability of our system to produce simple output, at no significant cost in meaning preservation. An example of output from our sentence fusion system, along with crowdsourced simplifications, can be seen in Table 1.1.

Our approach has several key advantages. Firstly, it is flexible, enabling the desired attributes of the created simplifications to be chosen. The user of the system can specify the desired balance between simplification and meaning preservation, which will then be respected by the system. As in the crowdsourced translation system created by Zaidan and Callison-Burch [2011], the user can choose to rely on crowdsourced human evaluations to select simplifications, which will result in a massive improvement in quality. Use of our ranker is still required if sentence fusion system sentences are to be included, although the ranker can serve as an intermediate stage with human annotators having the final word; we show that this hybrid system produces a significant improvement over only employing human simplifications.

As the sentence fusion component of our system is perhaps its largest innovation over the previous crowdsourcing research by Zaidan and Callison-Burch [2011], we begin with an overview of the history of sentence fusion methods and introduce the key techniques currently in use.

Chapter 2

Sentence Fusion

Sentence fusion is, at its most general, the act of taking a number of different sentences and creating a single sentences which combines their meaning. However, this definition is extremely broad, and without further information sentence fusion is not a well-defined task [Daume III and Marcu, 2004]. Unfortunately, human reviewers widely disagree about what information is important (and should thus be included in a fused sentence) and what information should be excluded. As a result, it is difficult to construct any human-driven, generic metric of sentence fusion quality [Daume III and Marcu, 2004].

This fundamental fact challenges or influences all other work in the field, but it does not mean that all work in sentence fusion is pointless. As Krahmer et al. [2008] find, generic sentence fusion is an ill-defined task precisely because it is generic. More directed sentence fusion tasks, such as query-driven sentence fusion, can yield better results [Krahmer et al., 2008]. There is equal reason to believe the same to be true in domains such as machine translation, where clear goals (i.e., ground-truth translations) are the norm.

Thus, despite the ambiguity at the heart of its existence, sentence fusion has seen a great deal of research over the past two decades. Numerous methods have been developed, varying from linguistics-based approaches which heavily emphasize grammatical structure to the use of modified machine translation systems. As would be expected from the insights of Krahmer et al. [2008], different contexts of sentence fusion are to some extent different tasks, and tend to inspire different approaches. Nonetheless, there are clear themes which recur across many domains, particularly the structuring of sentence fusion as a graph problem in which creating a fusion sentence requires finding a path through the graph, or finding a subgraph. Later on, we will also see that the use of machine translation systems themselves forms a common thread

Domain \ Approach	Graph-Based	Grammatical	Translation	Other
Summarization	[Filippova, 2010]	[Filippova and Strube, 2008] [Barzilay and McKeown, 2005] [Cheung and Penn, 2014]	[Rush et al., 2015] [Chopra et al., 2016] [Nallapati et al., 2016]	
Translation	[Bangalore et al., 2001] [Rosti et al., 2007b] [Sim et al., 2007]		[Zhou et al., 2017] [Firat et al., 2016a,b] [Zoph and Knight, 2016]	[Frederking and Nirenburg, 1994] [Ma and McKeown, 2015]

Table 2.1: The sentence fusion methods under consideration in this survey, divided by overall approach and domain. Graph-based and grammatical approaches to summarization are described in section 2.1.1, while the use of graph-based approaches to sentence fusion for machine translation is discussed in section 2.2.2. We discuss the usage of machine translation systems themselves for sentence fusion on machine translation-related tasks in section 2.2.3, and the use of machine translation-based sentence fusion for text summarization in section 2.3.

in much research, even when the tasks under consideration are not linked to translation. We will break down our examination of sentence fusion by both domain (what sentences are we fusing, and why?) and approach (how are we fusing them?). We will begin by examining some of the traditional methods for sentence fusion in text summarization, before continuing to consider the role of sentence fusion in machine translation. Finally, we will conclude with a return to text summarization, examining how developments from machine translation have impacted sentence fusion techniques for summarization.

2.1 Summarization

Text summarization, and the closely-related task of *text compression*, are perhaps the most obvious application of sentence fusion. In text summarization or compression, a large quantity of text (from one or more sources) is reduced to only a few sentences. For example, consider the case of Google News. Google News covers thousands of stories each day, and collects tens to hundreds of articles on each. To unify these articles, Google would like to be able to create a single description of each underlying news story (e.g., “John Edwards drops out of Democratic primary after extramarital affair revealed"), which could then be shown to users. In fact, this precise example motivates a noteworthy paper in sentence fusion conducted by a research scientist at Google, Filippova [2010].

A common approach to this task, which negates the need for sentence fusion, would be to directly *extract* text from the input we are given; in this case, it might take the form of choos-

ing one or more sentences from the collected news articles. Some reasonable methods could be used to do so; for example, one could attempt to group sentences by semantic similarity and choose the most representative sentence. In general, this set of solutions is often called *extractive text summarization*. However, this approach has historically tended to result in sentences which require substantial modification, as their meanings may not be appropriate outside of their original contexts [Gupta and Lehal, 2010].

An alternative to this, however, would be to attempt to create a new sentence that is better adapted to the summarization task than any of the individual input sentences. This falls under a different school of text summarization, *abstractive text summarization*, and faces different challenges; now, the core issue lies in finding ways to represent natural language to allow the creation of novel sentences [Gupta and Lehal, 2010].

2.1.1 Graph-Based and Grammatical Methods for Summarization

One approach to sentence fusion for text summarization or compression, which became popular largely in the early 2000s, is the use of sentence graphs. In effect, a group of input sentences can be represented as a single graph, where nodes are words and edges are relations between words in the input text. Various methods can then be used to determine the best candidate fusion sentence from the graph.

In the simplest forms of this approach, such as in the work of Filippova [2010], edges are simply adjacency, such that there is an edge from one word to another if the second word follows the first directly in at least one sentence. With a graph in this format, edge weights can be chosen such that the lightest path from a sentence start token to the end token is likely to be a good fusion, based largely on word transition frequency. An example fusion graph is shown in Figure 2.1. This approach was originally drawn from a similar but related task, paraphrase generation, where Barzilay and Lee [2003] introduced a very similar model. Although Filippova [2010] does not report results on any common test benchmark, human raters confirm that the output produced is generally grammatical and of good quality.

This approach can also be conducted with dependency trees, as pioneered by Barzilay and McKeown [2005]. In this implementation, a dependency tree (a graph where nodes are words and edges indicate that one word is linguistically dependent on another) are computed for each input sentence, and then the dependency trees of the input sentences are aligned. This allows the creation of a fusion dependency tree, which can be converted to a fused sentence. This

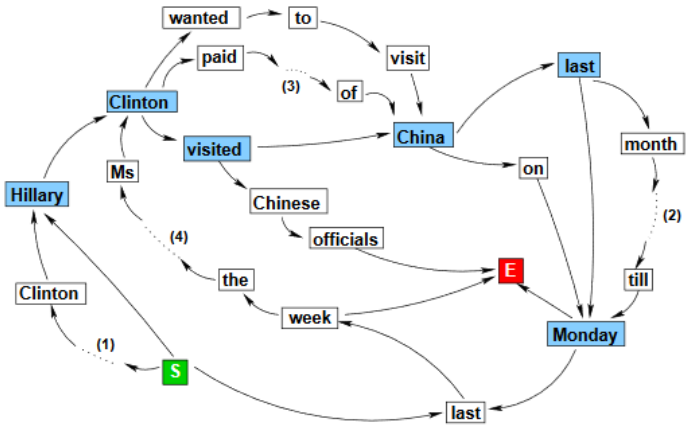

Figure 2.1: An example sentence graph reproduced from [Filippova, 2010]. The input sentences we wish to fuse have been transformed into a graph, where words are nodes and edges between words represent the flow of words in the input sentences. A fusion sentence can be created by finding a path between special start (S) and end (E) nodes. The nodes in blue demonstrate a possible fusion, "Hillary Clinton visited China last Monday."

technique was notably extended by Filippova and Strube [2008], who build off of the work of Barzilay and McKeown [2005] by adding an integer linear program model to create sentences from the combined dependency graph, after finding that the original methods were inadequate for German.

This approach was even further extended by Cheung and Penn [2014], who re-frame sentence fusion as "sentence enhancement". In effect, this is simply a change of focus; they are still using dependency-tree-based methods very similar to those of Filippova and Strube [2008], but they seek to use a richer set of semantic information to yield better results. They also demonstrate very good results in the news domain, showing improvement largely in the grammatical quality of their output.

2.2 Sentence Fusion in Machine Translation

Sentence fusion has repeatedly emerged as a theme in the field of machine translation, in several different contexts. First, sentence fusion was envisioned as a means of augmenting machine translation systems, by enabling their outputs to be combined; this spurred the development of

a number of sentence fusion schemes and showed good results in translation, but most of these sentence fusion models were broadly similar to those already in use elsewhere. Later, however, it became apparent that it would be desirable to have machine translation systems themselves be able to conduct sentence fusion, in order to incorporate multiple input sentences simultaneously, leading to substantial innovations worth considering. Before we begin, however, it will be useful to establish some terminology for discussing machine translation.

2.2.1 Types of Machine Translation

It is important to distinguish between two different types of machine translation systems: statistical, or *phrase-based*, machine translation systems (SMT), and neural machine translation (NMT) systems. Statistical machine translation systems operate by creating explicit, probabilistic mappings between words or phrases in two languages, and using this mapping to determine the most likely sentence in a target language corresponding to a sentence in a source language. These mappings, often called *phrase tables*, are easily examinable by the operator of the system, and thus make the system relatively transparent. Statistical machine translation systems have been in use for decades, and by the 2000s were well-established and reasonably standardized [Koehn et al., 2007].

Neural machine translation, on the other hand, is a much more recent approach, based on the use of large neural networks with relatively sophisticated architectures. Neural machine translation become particularly popular after Bahdanau et al. [2014] demonstrated revolutionary performance improvements, and has since become the de facto standard for machine translation, being quickly adopted by Google Translate [Wu et al., 2016]. Although neural machine translation systems are still fundamentally statistical in nature, their operations are harder to inspect for outside users; neural machine translation systems frequently possess tens to hundreds of millions of parameters, most of which are not directly identifiable with words in either the source language or the target language.

The precise details of the workings of neural machine translation systems are beyond the scope of this work, but a basic understanding will prove useful for later discussions of NMT. The neural machine translation system introduced by Bahdanau et al. [2014] consists of three key components, an *encoder*, which translates the input into a high-dimensional numerical representation, a *decoder*, which produces text in the target language given an input high-dimensional numerical representation, and an attention mechanism, which is novel in Bahdanau et al. [2014]

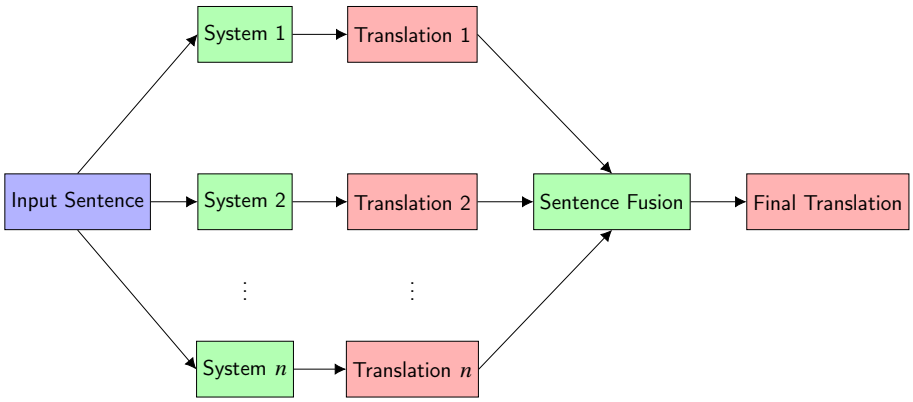

Figure 2.2: A generalized diagram of how system output combination functions. n different machine translation systems are applied to an input sentence in parallel, and their outputs are then combined into a single output sentence using a sentence fusion system. The methods considered differ only in terms of how the sentence fusion step itself is conducted[Rosti et al., 2007b; Sim et al., 2007; Zhou et al., 2017; Ma and McKeown, 2015].

and enables the decoder to examine only those parts of the encoder's output which are relevant to the part of the sentence it is currently translating. This basic structure will persist in all of our discussions of NMT, but some of the papers under consideration introduce variations.

2.2.2 System Combination for Machine Translation

Sentence fusion has also been considered from the standpoint of combining the outputs of multiple machine translation systems. This is a natural consequence of the fact that it is fairly easy to create multiple machine translation systems with different strengths and weaknesses, and that combining their outputs (in an intelligent way) could allow for a composite system stronger than any of its individual components. This approach was initially inspired by strong results from composite systems in speech recognition, and research into it began even in the early days of machine translation. A foundation was laid by Frederking and Nirenburg [1994], who take the approach of merging monotonically aligned translations. They are able to improve their results by employing estimates for the quality of each component of each sentence, which are generally produced automatically by statistical machine translation tools. This initial approach showed some promise, but it was heavily limited by the fact that the output of different machine translation systems systems frequently cannot be monotonically aligned.

To circumvent this problem, many researchers found ways to avoid explicitly fusing sentences entirely. For example, Rosti et al. [2007a] find that they are able to directly create a composite machine translation system from many individual systems without actually merging output sentences. This approach, however, is dependent on the use of statistical machine translation systems, as it directly modifies their phrase tables, and cannot straightforwardly be adapted to neural machine translation.

Thus, a more general solution is needed, one which can also accommodate neural machine translation systems – and this comes at the cost of returning to sentence fusion as a task. After all, if translation systems are treated as black boxes, then their outputs must be combined *after* it has been produced, which requires sentence fusion. Development of such general methods began early, with the work of Bangalore et al. [2001]. Bangalore et al. [2001] used simple graph-based methods reminiscent of those used by Filippova [2010], where output sentences from machine translation systems (referred to as "hypotheses") are combined into a directed acyclic graph of words. As in Filippova [2010], the task of creating an output sentence can then be reduced to finding the best path through the graph. Bangalore et al. [2001] consider various metrics for doing so, such as simply choosing the most common path, or employing a language model to rank paths by the fluency of their resulting sentences. Although the approach of Bangalore et al. [2001] is much narrower than that of Fillipova, it still demonstrated an ability to produce output superior to that of the best translation system it was based on.

The graph-based approach of Bangalore et al. [2001], which has come to be known as "consensus networks", has proved a fertile ground for further research. For example, both Rosti et al. [2007b] and Sim et al. [2007] develop methods to determine the best possible path through a consensus networks, and show improved performance on the machine translation metric BLEU on translation benchmarking datasets. Rosti et al. [2007b] includes improved statistical methods for path choice as does Sim et al. [2007]. Interestingly, however, Sim et al. [2007] also test a method in which they refuse to conduct any word-level sentence fusion at all; instead, they rank the hypotheses by their similarity to what *would* have been the chosen path, and choose the most similar hypothesis. This effectively chooses the median hypothesis, and has the advantage of not breaking up phrases, and was found to produce the best results of all tested methods on a machine translation benchmark. In effect, we can think of this as a reprise of extractive text summarization methods (as no new sentence is created), but applied in machine translation and using a sentence fusion system as an extraction tool.

Other, non-graph-based methods have also been considered. For example, Ma and McKe-

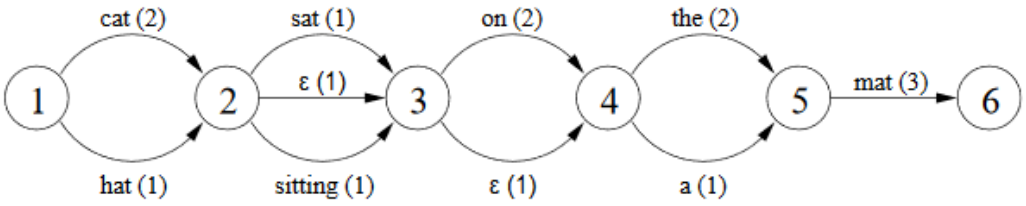

Figure 2.3: An example consensus network, reproduced from Rosti et al. [2007b]. Three possible translations are equally likely, "cat sat on the mat", "cat on the mat" and "cat sitting on the mat". The consensus networks generated by Sim et al. [2007] and Bangalore et al. [2001] are visually very similar.

own [2015] augment graph-based methods similar to those of Bangalore et al. [2001] by introducing a paraphrase-based model, extracting paraphrase likelihoods from aligned monolingual (i.e., target language) data. This paraphrase model is used in parallel with the more traditional graph-based method to generate a set of candidate output sentences, from which a consensus sentence is chosen based on its (weighted) similarity to other sentences. Individually, the paraphrase-based method is only at best comparable to previous methods, but it produces a substantial boost on Chinese-English and Arabic-English benchmarks when used in combination with graph-based approaches.

Even more radical departures from the graph-based approaches of the past exist, however. Neural machine translation systems themselves have been considered as a tool for system output combination. Zhou et al. [2017] modified the classic neural machine translation architecture to permit the simultaneous use of multiple inputs, by creating a unique encoder and attention mechanism for each input, and then adding an additional attention mechanism which determines which of the inputs the decoder will use. When used for machine translation system combination, this showed strong improvements in translation quality provided that the systems used as input are not too correlated, which is likely to occur in the case of combining outputs from statistical and neural machine translation systems. This is, however, not a general neural sentence fusion system, as it is dependent on learning the relative strengths of each system (Zhou et al. [2017] in fact cite this as a strength), and does not automatically generalize out to an arbitrary (and varying) number of input sentences.

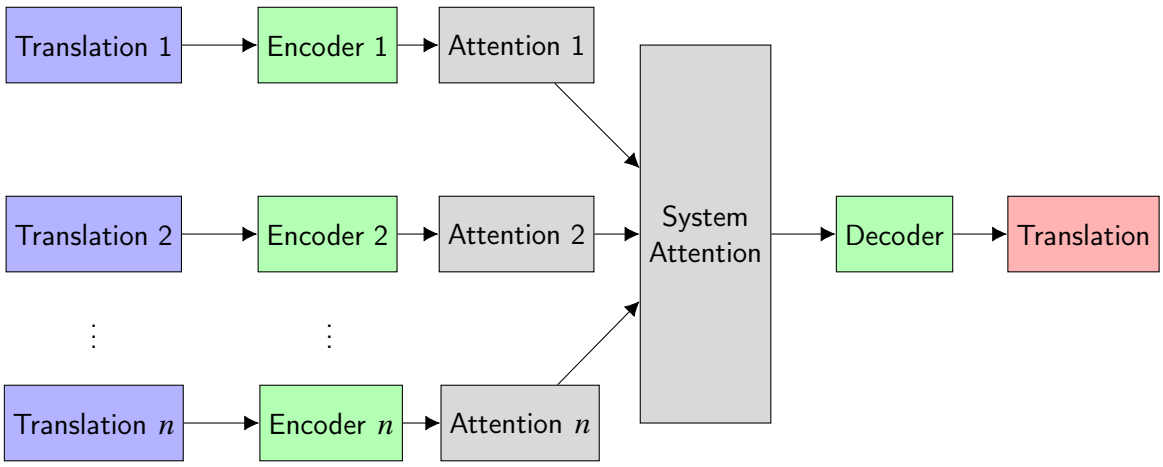

Figure 2.4: A neural sentence fusion model, as introduced by Zhou et al. [2017]. Here, n different systems (not shown) are translating a single source sentence. Their outputs are then given to the neural sentence fusion model, which reads in each as though it attempting to translate from each of them simultaneously. This would occupy the slot of sentence fusion model in Figure 2.2

2.2.3 Sentence Fusion During Translation

In the previous examples, sentence fusion was a step conducted after machine translation systems had already acted. However, there are also situations where it would be desirable to create machine translation systems themselves capable of directly conducting sentence fusion. For example, consider the case of exploiting data from multiple languages; if multiple source sentences are known (for example, in a case where parallel French and English versions of a document exist, and the target language is German), then they may help to disambiguate each other. In practice, however, cross-linguistic versions are rarely perfectly-literal translations of each other, making this is a sentence fusion task, as decisions must be made about what content is kept and what content is dropped.

Of course, it is possible to exploit data from multiple languages without explicitly conducting sentence fusion, but such methods are limited. Some progress can be made by exploiting the transparency of statistical machine translation, as by Cohn and Lapata [2007], but this is not obviously relevant in an era when neural machine translation dominates. Alternative methods for leveraging cross-linguistic data with neural machine translation without conducting sentence fusion are largely based on creating systems capable of translating between different languages

pairs, considered one at a time, as by Google in Johnson et al. [2016]. Although this approach has yielded substantial benefits, it is still limited, as it would be desirable to simultaneously employ multiple source sentences in different languages when they are available [1].

Fortunately, methods for doing so have been created by multiple research groups. A system fully capable of conducting translation using source sentences in different languages was introduced by Zoph and Knight [2016], while Firat et al. [2016a,b] introduced a system capable of translating between many different language pairs, and then extended it to enable the simultaneous use of multiple source sentences. Both systems are roughly comparable; the core difference between them lies in how they merge information from the multiple input sentences. In brief, Zoph and Knight [2016] introduce a novel combination unit which serves to merge the outputs from different encoders, while Firat et al. [2016b] consider either explicitly averaging information at either the input to the decoder, or training two separate models and averaging their predictions at a word-by-word level. Both systems show promise in improving results when parallel source text is available.

2.3 Translation and Summarization Together

The potential of neural machine translation systems for sentence fusion has led researchers to consider their use in text summarization. This is helped by the fact that summarization can, with relatively little difficulty, be rephrased as a machine translation problem; given a set of input sentences, likely in sequence, produce ("translate" to) an output sentence or sentences which capture its meaning. Initial work on applying neural machine translation to abstractive text summarization was carried out by Rush et al. [2015], who apply modified forms of neural machine translation systems to text simplification. Interestingly, however, they choose not to use the standard formulation of neural machine translation system laid out by Bahdanau et al. [2014]. Instead, they replace the encoder defined by Sutskever et al. [2014], considering several candidate encoding systems rooted in more traditional natural language processing approaches, such as "bag of words"-based encoding, in which only the identities of the words present are considered and word order is ignored. They also use a non-standard type of decoder, although they apply an attention model very similar to that of Bahdanau et al. [2014].

[1] It is not difficult to see how this might be useful even in the context of Google Translate; for example, I am fluent in English and French, and often use Google Translate to query sentences in Chinese. I would not mind the extra effort of formulating my query in both English and French, if it resulted in an improved translation.

Furthermore, using neural networks creates a new challenge: neural networks generally require exceptionally large amounts of data to train their hundreds of thousands or millions of parameters, while graph-based approaches do not, as there are often nearly parameter-free. To solve this, Rush et al. [2015] create a large training corpus from publicly available news articles, using the first sentence of each article as the target summary and the rest of the article as input. With this done, however, they find that their performance is substantially better than that of the previous best non-neural systems on traditional benchmark text summarization datasets. For comparison, they also test more traditional statistical machine translation systems, and find that the neural model is significantly superior.

After the work of Rush et al. [2015], the obvious next step was to directly employ the standard neural machine translation framework, as even Rush et al allude to. It did not take long for work on this topic to appear, with Chopra et al. [2016] directly employing a standard neural machine translation system on the dataset used by Rush et al. [2015] and producing superior output. This was further extended by Nallapati et al. [2016], who augment their encoder by providing information about part-of-speech (i.e., noun, verb, adjective, etc) and relative word frequency, in addition to the word itself. This serves, in their phrasing, to guide the production of a fusion summarization sentence by helping identify key words, which are likely to be nouns that occur much more often in the input text than in ordinary English. Finally, they also include a mechanism for gracefully handling rare words which are not known to the system, exploiting the attention mechanism to allow the system to directly copy unknown words to the output. With these modifications in hand, they again report substantial improvements over Rush et al. [2015] and Chopra et al. [2016] on the same datasets.

It is important to note, however, that the abstractive text summarization methods shown here are not truly general-purpose sentence-fusion systems. They are formulated to assume that their input is a single stream of text, flowing logically as in a single article, while in truly general sentence fusion multiple *parallel* texts should be available as input. In some other respects, however, they show remarkable potential to surpass even traditional sentence fusion systems; in particular, the work of Nallapati et al. [2016] allows for the production of multiple-sentence outputs. If this trend continues, it could eventually allow sentence fusion to broaden into paragraph or even document fusion, creating entirely new applications.

2.4 Summary

Sentence fusion methods have evolved greatly over the past decade, repeatedly re-emerging to confront new challenges as the need for sentence fusion appears in different domains. Although graph-based approaches have historically been dominant, especially in areas related to text summarization, developments in neural machine translation have recently provided a new avenue forward, creating entirely new systems bearing little resemblance to previous methods. Indeed, given the numerous successes of neural models in performing sentence fusion in various specific contexts, such as the work of Zoph and Knight [2016] for multi-source translation and Nallapati et al. [2016] for abstractive text summarization, it might be tempting to say that a truly general neural sentence fusion system is just around the corner. Nevertheless, the message of Daume III and Marcu [2004] should not be forgotten — generic sentence fusion is, in fact, not a well-defined task, and any neural model ultimately created will have to exist in the context of a smaller, better-defined subtask of sentence fusion.

Chapter 3

Methods

Having introduced sentence fusion, we will now discuss our approach to crowdsourcing text simplification with sentence fusion. We will begin with an overview of the domain, including our data and evaluation metrics, before proceeding to discuss how we gathered crowdsourced human simplifications. We will then introduce our sentence fusion system, which is modeled on that of Filippova [2010], and discuss how we choose between

3.1 Data

We used data from the Newsela dataset Xu et al. [2015] as our source data. This dataset consists of a corpus of 56,037 sentences from news articles, along with their corresponding simplifications. Each article was hand-simplified by experts to one of four different levels (referred to as V1, V2, V3 and V4, in order of increasing simplicity). Each original sentence in the corpus is aligned with one or more of these simplified sentences, and alignments between simplified sentences at different levels of simplification are also provided.

To conduct our initial training, we used a set of 119 original sentences, drawn randomly. For our testing set, we chose 200 other random sentences, separately from the training set, subject to the restriction that each was aligned with a sentence of each level of simplicity in the Newsela corpus. Furthermore, in the case that any chosen original sentence was aligned to multiple simplified sentences at a given simplification level (representing a case of sentence-splitting), these sentences were appended together, creating a single multi-sentence simplification.

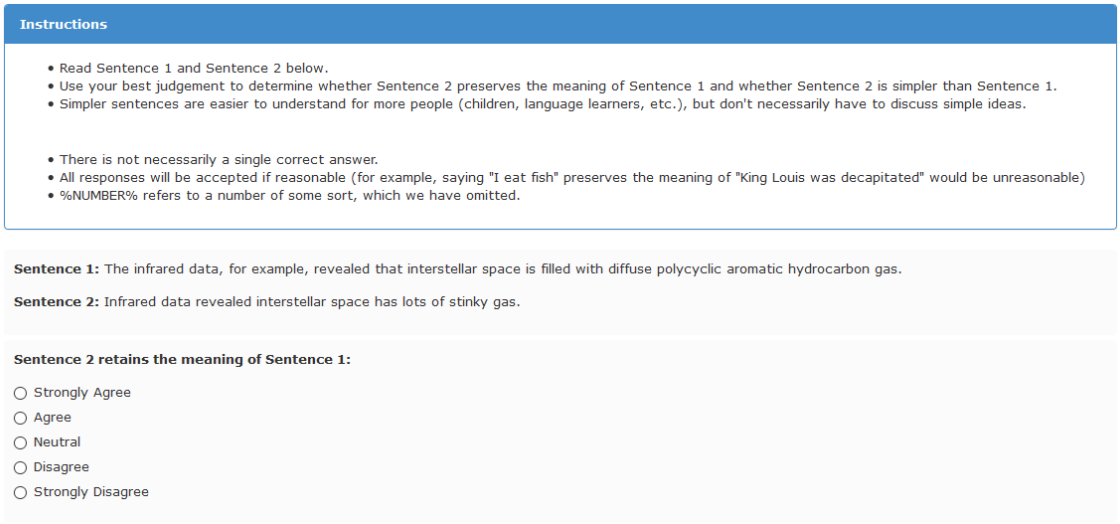

Figure 3.1: An example prompt used to solicit judgments of adequacy. The simplicity prompt was similar.

3.2 Evaluation

We assessed each simplification using human evaluation, across two different metrics, **simplicity** and **adequacy**, both of which are standard metrics for human evaluation for text simplification [Xu et al., 2016]. Simplicity is defined as the change in simplicity between a simplification and its corresponding original sentence, and was solicited using the prompt "How much simpler is sentence 2 than sentence 1". Simplicity was assessed on a five-point scale ranging from -2 (much less simple) to 2 (much simpler).

Adequacy is defined as the degree to which a simplification preserves the meaning of its corresponding original sentence, and was solicited using the Likert-scale prompt "Sentence 2 preserves the meaning of sentence 1". It was assessed on a five-point scale ranging from 1 (strongly disagree) to 5 (strongly agree). An example prompt for adequacy is visible in Fig. 3.1

For each metric and for each sentence, we collected annotations from three separate human annotators on Amazon Mechanical Turk. These annotations were then averaged together.

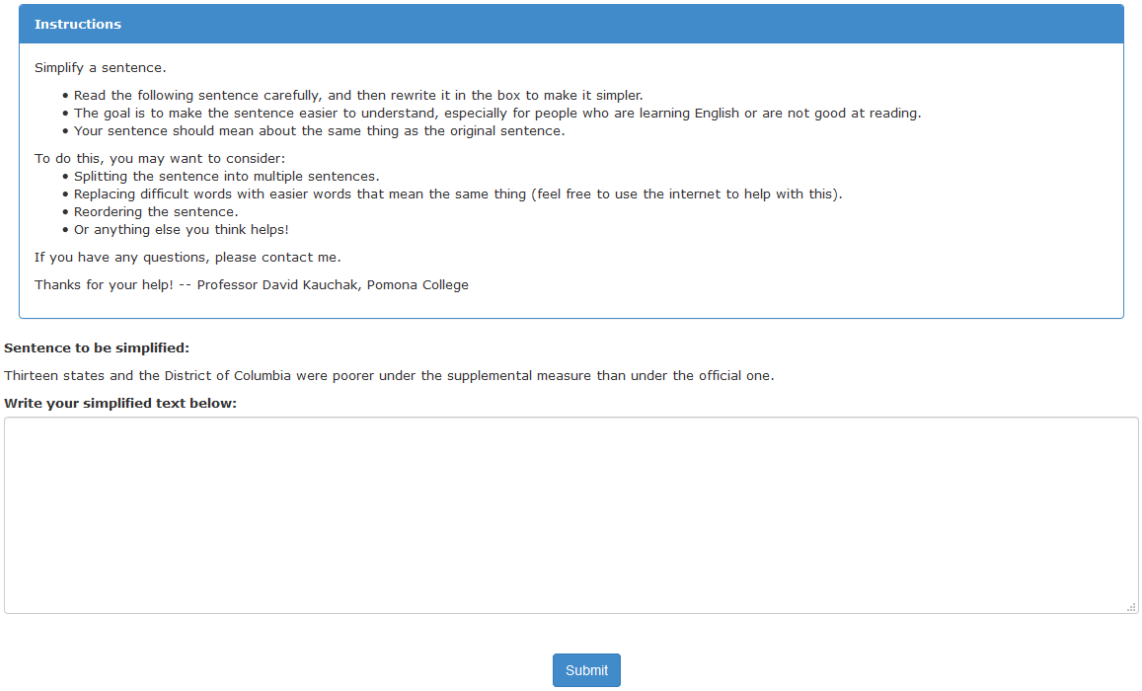

Figure 3.2: An example prompt used to gather human text simplifications from workers on Amazon Mechanical Turk.

3.3 Human Simplification

The core of our approach is composed of gathering a number of sentences from inexpert reviewers, via Amazon Mechanical Turk (an example prompt can be seen in Fig 3.2). We had four humans simplify each sentence in our testing and training sets, yielding a total of $319 \cdot 4 = 1276$ human simplifications gathered. Users were required to be in the United States (to ensure English proficiency), and to have a historical 97% task acceptance rate and at least one previously accepted task, to prevent bots or otherwise undesirable workers from attempting our task. However, no requirements were placed on education level, English proficiency or previous simplification experience. This was done intentionally; our goal was to demonstrate that true amateurs can perform simplification excellently.

3.4 Sentence Fusion

To perform sentence fusion, we used a graph-based sentence fusion system derived from the work of Filippova [2010], which was briefly introduced in Section 2.1.1. We will now describe this sentence fusion system in more detail, before proceeding to introduce our modifications to the model.

The sentence fusion system designed by Filippova [2010] is based on the creation of word graphs, in which sentence fusion is performed by finding paths. In Fig. 3.3, we can see an example of such a graph, created from the sentences The words highlighted in blue represent the shortest path through the graph, which would create the sentence "Hillary Clinton visited China last Monday". This graph was created from the following sentences:

- The wife of a former U.S. president Bill Clinton Hillary Clinton visited China last Monday.
- Hillary Clinton wanted to visit China last month but postponed her plans till Monday last week.
- Hillary Clinton paid a visit to the People's Republic of China on Monday.
- Last week the Secretary of State Ms. Clinton visited Chinese officials.

This graph would lead to the immediate creation of the fusion sentence "Hillary Clinton visited China last Monday," which is an example of a simplifying sentence we would seek to be able to create.

In this section, we will walk through the creation of these graphs, and then introduce the mechanisms used to find paths in the graphs. First, we will discuss the graphs themselves. A sentence fusion graph (henceforth referred to simply as a graph) is directed graph, consisting of a tuple of nodes and edges (V, E) produced from a set of input strings $W = \{w_1, w_2, \ldots, w_n\}$. Each entry in V represents either a word or punctuation, or a special token, one of "START" (a unique node marking the start of a path) or "STOP" (a unique node marking the end of a path). Each element in V representing a word corresponds to at least one word in some input string w_i, and possibly many. For any two nodes v_n and v_m which represent words or punctuation, an edge from v_n to v_m will exist in E if and only if, in some input string w_i, we have that v_n corresponds to a word immediately before a word corresponded to by v_m. Thus, edges can be thought of as word adjacencies. There will also be edges from the special "START" token to the

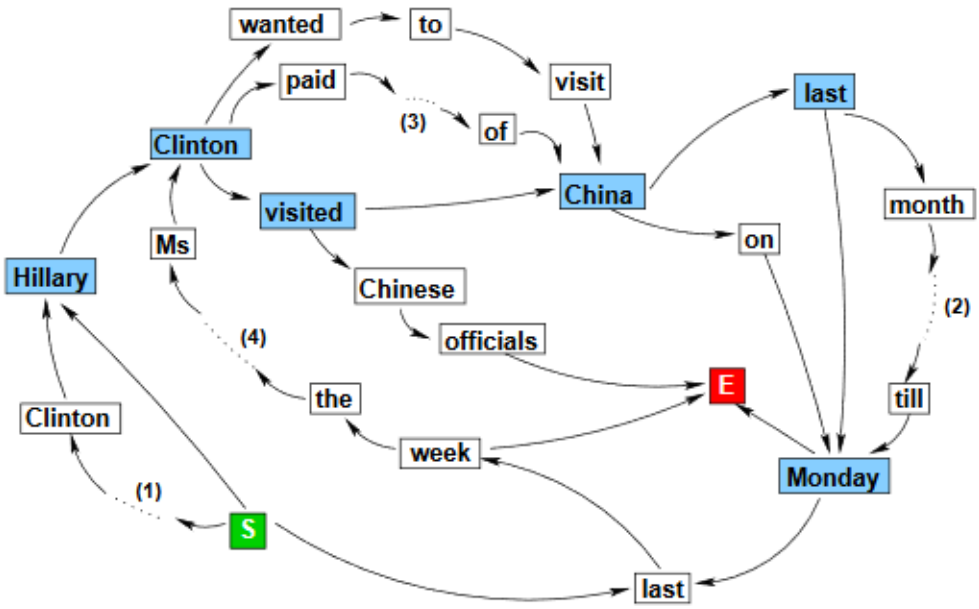

Figure 3.3: An example graph created by the original sentence fusion system of Filippova [2010], drawn from their paper.

nodes representing the first word of each input string, and edges from the nodes corresponding to the last word of each input string to the "STOP" token. In general, we chose to implement these graphs using the Python graph library NetworkX [Hagberg et al., 2008].

Stopwords

Following the precedent set by Filippova [2010], we chose to treat stopwords (extremely common words such as "the", "and" or "at") differently from other words. Words flagged as stopwords were treated differently both in feature generation (see Section 3.5.1) and in graph creation (see below). To identify stopwords, we used the list of stopwords included in NLTK [Loper and Bird, 2002], augmented with all punctuation.

String Preprocessing

Each word was annotated with its part of speech, and words were represented as (string, POS) tuples. This serves to avoid errors such as "uniform" as a noun and "uniform" as an adjective being taken to be the same word; it would be desirable, in such cases, for our system to treat these words as separate. To conduct part-of-speech tagging, we used the utility function included with NLTK [Loper and Bird, 2002]. We also segmented input strings into sentences using the punkt tokenizer included with NLTK; this tokenizer is pre-trained on an English corpus, so it could be directly used. We inserted an "EOS" token between separate sentences in each string. This was useful both for handling capitalization, and for interfacing with our neural language model (see Section 3.5.1), which expected such tags to be present.

3.4.1 Graph Creation

Having introduced the basic structure of a graph, we will now discuss how graphs are created. Graph creation is an iterative process, with input strings from W added to the graph one at a time. In our work, the first string added to the graph always represents the original sentence, as this is necessary for our alignment methods (see Section 3.4.1). Each word from this first string will be mapped to a new node in the graph, and edges will be created between all nodes representing adjacent words. We will also insert nodes representing our "START" token and "END" token and create an edge from the "START" token to the node representing the first word of the initial string, and an edge from the node representing the final word of the string to the "END" token.

Next, the remaining strings to be merged are added. When a new string is added, the sentence is processed iteratively, one word at a time. For each word, if there are no possibly-corresponding nodes (i.e., this word was not in any of the preceding strings), we create a new node. Otherwise, all non-stopwords which correspond to nodes already in the graph will be matched with these nodes. In cases where multiple nodes exist in the graph, all of which could correspond to a given non-stopword, we choose the node whose neighborhood overlaps the most with the words around the current word in the string under consideration. For stopwords, we will create a new node unless one of the corresponding nodes in the graph shares context (i.e., a preceding or succeeding word) with the current stopword. Next, we create an edge from the previous node (if this is not the first word) or the "START" token (if the was the first word) to the chosen node. If this was the last word in the string, we will additionally create an edge from this word to the final "STOP" token. This process is then repeated for each string.

Alignment Variant

Empirically, we observed that many nonsensical synthetic sentences were being created from the system described above, sometimes to such a degree that it was difficult to extract reasonable synthetic sentences. To resolve this, we introduced an additional alignment step, which served to reduce the total number of paths through the graph. We used the Berkeley aligner [Liang et al., 2006] to create word alignments between our human simplifications and the original sentence. These alignments are used when adding a sentence to a graph; we merge a non-stopword with an existing node if and only if our alignment indicates that the word was aligned with its counterpart in the original sentence. This has the effect of reducing the number of paths through the graph, and we found it to have a substantial qualitative improvement.

3.4.2 Path Creation

Once a graph was created, we create sentences from it by finding paths from the "START" token node to the "END" token node. Although in principle it would be possible to find all such paths (so long as no cycle exists in the graph), this would impose an unreasonable computational cost, as some reranking features (see below), particularly our neural language model (see Section 3.5.1), are quite computationally costly. As a result, we chose to generate only a limited number of paths from each graph, imposing a cap of 1000. However, this introduces the risk that the first 1000 paths generated will all be of low quality, while a potentially high-quality path might be

missed as it lies after our threshold.

To address this, we attempt to eliminate as many low-quality sentences as possible prior to reranking, while finding paths from the graph in an order that corresponds to their quality. We began by, heuristically, finding paths in the order of shortest-to-longest, again using NetworkX. This has the advantage of ignoring fusion paths that simply consist of combining multiple sentences, which might otherwise compose a large fraction of our output. We augmented this by introducing the edge-weighting scheme described by Filippova [2010]. In this system, a weight is associated with each edge in the graph; these weights are used in finding shortest paths. For an edge $e_{i,j}$, Filippova [2010] defined the weight $w(e_{i,j})$ as:

$$w(e_{i,j}) = \frac{freq(i) + freq(j)}{\sum_s \frac{1}{diff(s,i,j)}},$$

where $freq(w)$ is the number of times that word occurred in all of the strings used to create the graph, $\sum_s$ is a sum over the strings used to create the graph, and where $diff(s,i,j)$ is defined as

$$diff(s,i,j) = \begin{cases} pos(s,i) - pos(s,j) & pos(s,i) > pos(s,j) \\ 0 & \text{Otherwise} \end{cases}$$

where $pos(s,i)$ is defined as the index of word i in string s.

This shortest-paths system, following the heuristics of Filippova [2010], has the advantage of encouraging a shortest-paths search to find reasonable paths through the graph that respect the original meanings of the sentences involved.

In addition to this, we also performed filtering, to further eliminate extraneous sentence fusion candidates prior to including them in our reranking system (see below). To do this filtering, we limited ourselves to considering two attributes, the compression ratio (excluding stopwords) between the original sentence and each sentence fusion candidate, and the Siamese LSTM similarity measure between a sentence fusion output sentence and the original sentences. To calibrate our length difference, we examined the distribution of compression rations among human simplifications in our training set, and found a mean of 0.897, with a standard deviation of 0.226. We chose to use a filtering window of one standard deviation, resulting in all sentence fusion candidates with non-stopword compression ratios outside of the interval $(0.671, 1.12)$ being rejecting.

Meanwhile, to filter using Siamese LSTM similarity, we chose to use a dynamic threshold based on the similarities of the human simplifications in use. For each graph, we used the mean Siamese LSTM similarity between the human simplifications used to create a graph and the original sentence as a threshold, meaning that only system fusion sentences more similar than this threshold were included in our reranking. This, again, had the advantage of removing sentence fusion candidates that deviated too much from the original sentence prior to inclusion in our reranking.

3.5 Ranking

The sentence fusion system laid out in section 3.4 is capable of generating an extremely large number of hypothesis sentences of widely varying quality. Moreover, the multiple sentences produced by annotators (as described in section 3.3) are also of varying quality. As the goal of our system is to recommend a single output sentence for each original human input sentence, we thus create a method for selecting the "best" possible output sentence from those available, for some criterion.

To do this, we employ a reranking algorithm based on linear-kernel support vector machines [Herbrich et al., 1999; Lee and Lin, 2014], trained to rank sentences to maximize some combination of adequacy and simplicity (see section 3.2). This system transforms ranking into a pairwise classification problem, which is tractable for a traditional support vector machine; we employ an implementation provided by Pedregosa et al. [2012].

3.5.1 Language Model

As our first feature, we employed a neural language model. We used the language model architecture from Kim et al. [2016], trained on the billion-word language model corpus developed by Chelba et al. [2013]. This neural language model was trained for three days using a GTX 1070 GPU, following the methods of Kim et al. [2016]. We used three hidden LSTM layers and three highway layers, with a vocabulary of 60,000 words, and attained a perplexity of 35 on the provided testing data. We worked in Keras [Chollet et al., 2015] and modified a publicly-available implementation[1] for use with datasets too large to fit into memory, such as the billion-word corpus.

[1] https://github.com/jarfo/kchar

To evaluate candidate sentences using the language model, we computed the negative logarithmic probability of each sentence, denoted as

$$W = (w_1, w_2, \ldots, w_n),$$

measured as

$$NLL = -\sum_{i=1}^{n} log p(w_i | w_1 w_2 \ldots w_{n-1})$$

However, NLL is not an ideal measure of the fluency of a sentence, as it estimates the total probability of a sentence, and thus inherently biases sentences toward brevity. A more appropriate measure might be the perplexity of each sentence, defined as $P = 2^{NLL/n}$, which is a measure of the "average" probability of a sentence across its length. This formulation, however, was found to reward sentences with more stopwords than necessary, and to cause numerical instability, as it can have extremely large values. Empirically, we found better success employing the averaged NLL formulation $P' = \frac{NLL}{l} n'$, where n' is the number of non-stopwords in sentence W (see Section 3.4).

3.5.2 Siamese LSTM

To measure sentence similarity, we used a Siamese LSTM system [Mueller and Thyagarajan, 2016]. This consists of two LSTM recurrent neural networks [Hochreiter and Schmidhuber, 1997] which process two sentences in parallel. These LSTMs have shared weights, allowing their hidden states to be compared after both sentences have been processed, directly yielding a measure of the similarity of the two sentences. We trained our model following the methods of Mueller and Thyagarajan [2016], on the same data that they employed. Note that as we used the Siamese LSTM to filter output of our sentence fusion system, measures of its utility calculated across our training set (as in Figs. 3.6 and 3.7, and Table 3.1) may understate its value.

3.5.3 Ngram Language Model

We also incorporated an ngram-based language model, employing the SRILM toolkit Stolcke [2002]. We trained a 3-gram language model using Kneser-Ney smoothing [Kneser and Ney, 1995] on the billion-word language model corpus developed by Chelba et al. [2013]. We directly incorporated the probability estimates from the 3-gram language model as a feature in our reranker, in the form of average logprob (defined as in Section 3.5.1) over all tokens in the

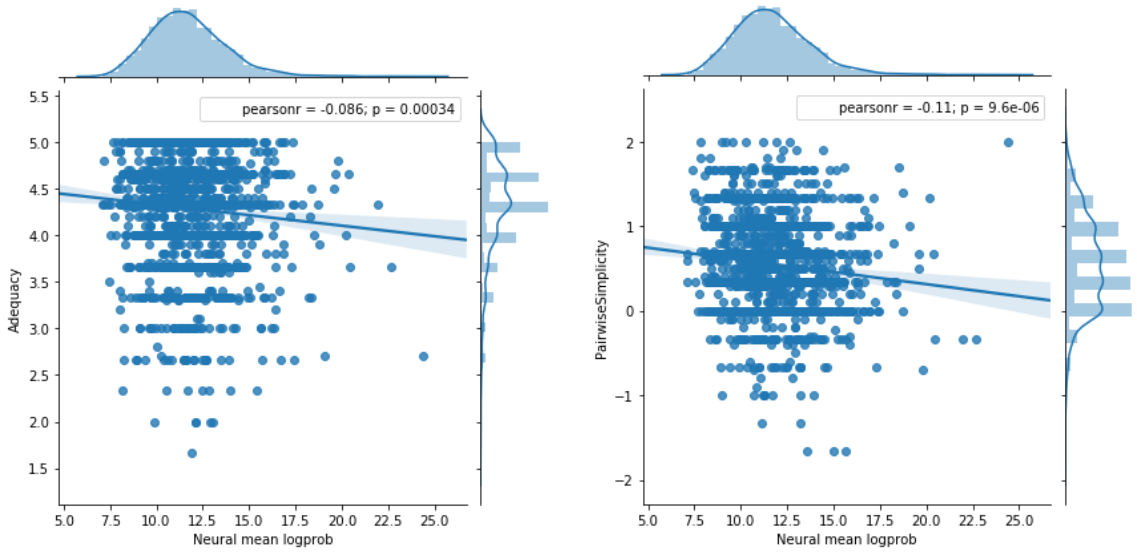

Figure 3.4: These plots analyze the importance of the neural language model as a feature. Note that a higher logprob indicates a less-likely sentence. Logprob has negative correlations with both simplicity and adequacy, making it particularly useful. Correlations calculated on training set.

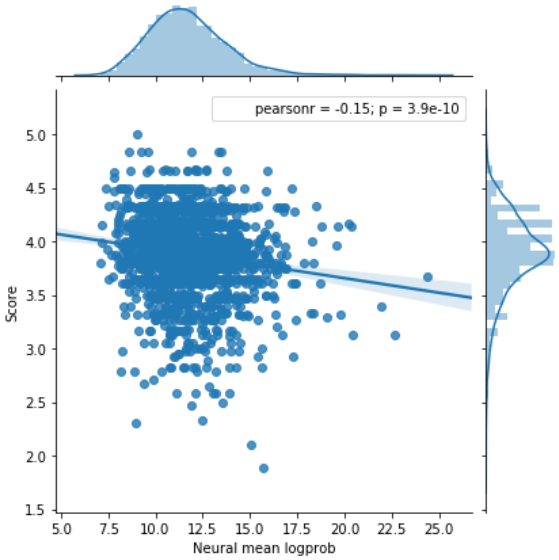

Figure 3.5: This plot shows the importance of the neural language model as a feature. Note that a higher logprob indicates a less-likely sentence. Logprob has a very significant negative correlation with score, making it particularly useful. Correlations calculated on training set.

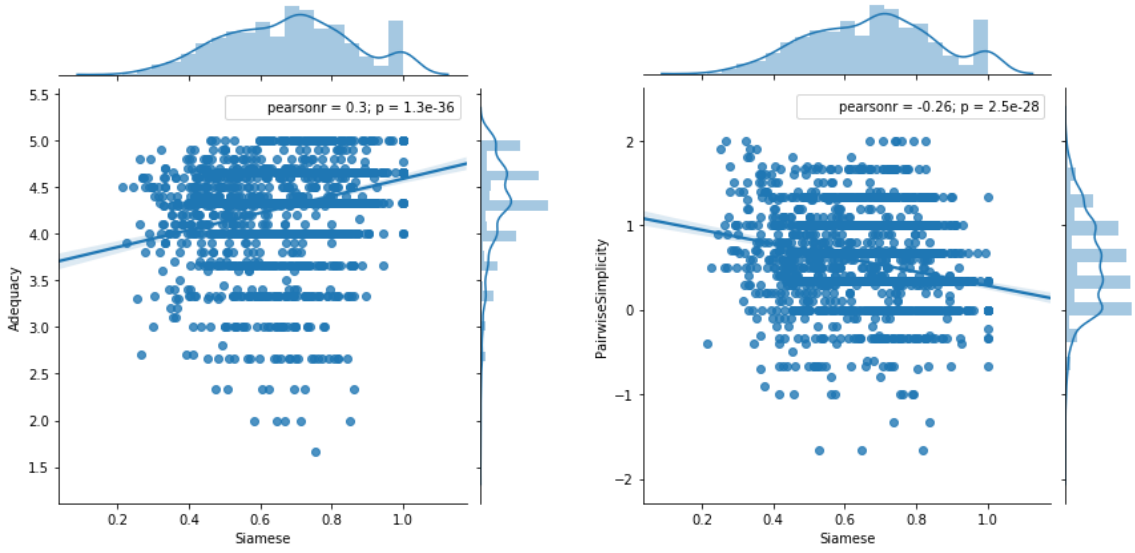

Figure 3.6: These plots analyze the importance of the Siamese LSTM model as a feature. Note that it has a strong positive correlation with simplicity while negatively correlating with adequacy. Correlations calculated on training set.

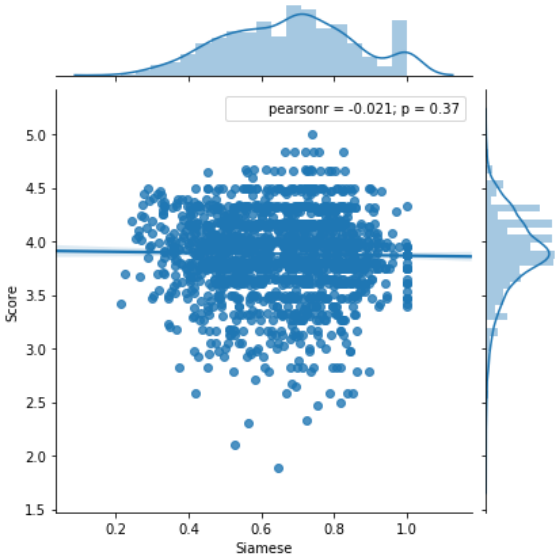

Figure 3.7: This plot demonstrates the importance of the Siamese LSTM model as a feature. Note that it has a only a limited correlation with score (defined as the geometric mean of adequacy and simplicity), so its value is questionable, although we confirm in Table 3.1 that its inclusion is apparently beneficial. Correlations calculated on training set.

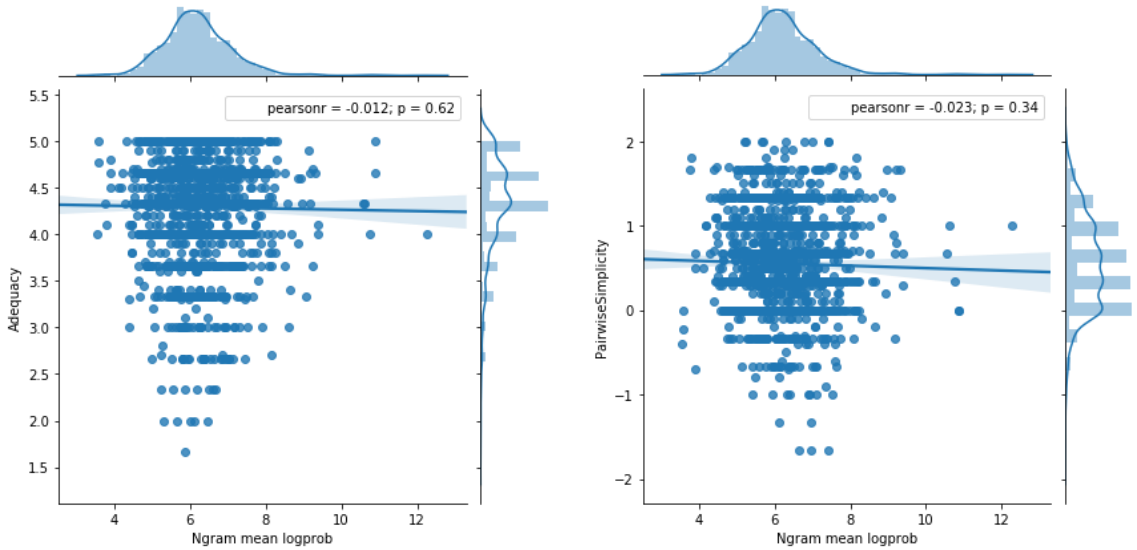

Figure 3.8: These plots analyze the importance of the ngram language model as a feature. The assessments of the Ngram model only weakly correlate with simplicity and adequacy. Correlations calculated on training set.

input sentence. Figs. 3.8 and 3.9 show that the ngram model is only a weak predictor of overall simplification quality, although we find in Table 3.1 that its inclusion is still beneficial.

3.5.4 TFIDF Bag of Words

To measure similarity between a sentence and its original counterpart, we used a bag-of-words similarity measure, weighted by TFIDF (term frequency-inverse document frequency). In the bag-of-words model, each sentence is represented as a multiset of the words present in the sentence, with all grammatical order discarded.

This model was weighted by TFIDF, which serves to increase the importance of uncommon words [Sparck Jones, 1972]. TFIDF is calculated at a word-by-word level as the product of the number of times a specific word occurs in a sentence (term frequency) and the inverse of the number of times that term occurs in all sentences in a corpus (inverse document frequency). We used Scikit-Learn to calculate TFIDF, which uses a slightly modified form of IDF:

$$\text{IDF} = log(\frac{n_d}{1+df(t,d)}) + 1,$$

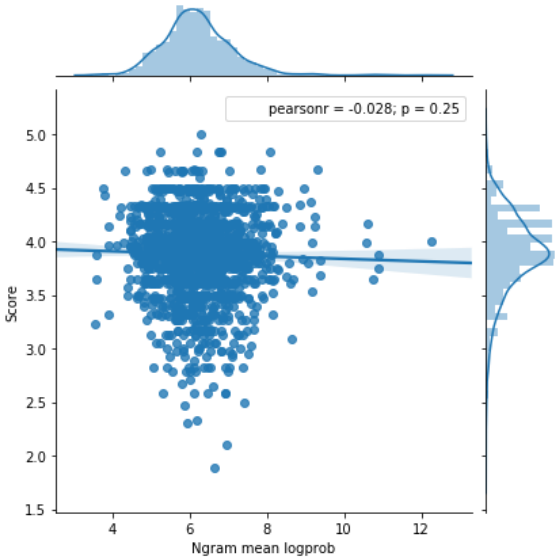

Figure 3.9: This plot demonstrates the importance of the ngram language model as a feature. Note that it has a only a limited correlation with score (defined as the geometric mean of adequacy and simplicity), so its value is questionable, although we confirm in Table 3.1 that its inclusion is apparently beneficial. Correlations calculated on training set.

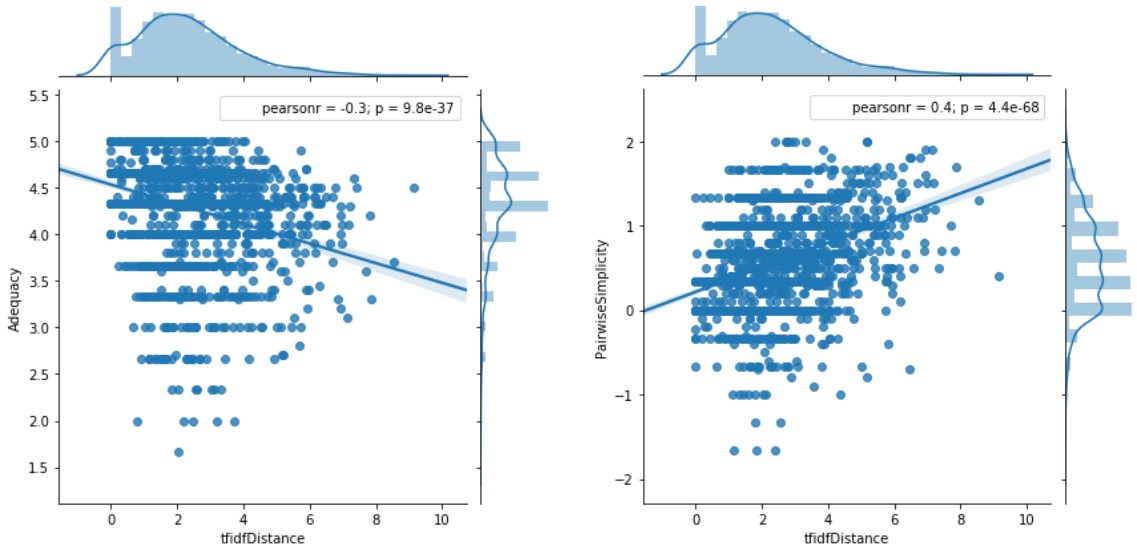

Figure 3.10: These plots analyze the importance of TFIDF as a metric. Note that it has a strong positive correlation with simplicity while negatively correlating with adequacy. Correlations calculated on training set.

where n_d is the number of documents in question and $df(t,d)$ is the number of documents containing term t; the addition of 1 serves to prevent divisions by zero. Scikit-learn also normalizes the magnitude of each vector using the Euclidean norm [Pedregosa et al., 2011].

We calculated inverse document frequency based on the entire Newsela corpus [Xu et al., 2015], treating every sentence in the corpus as an independent document for the purposes of calculating IDF, and then applied it to produce a weighted bag-of-words multiset for each sentence under consideration. This multiset was compared to the corresponding multiset for the original sentence using the cosine distance metric, defined as

$$C(A,B) = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}.$$

This was found to correlate strongly positively with simplicity and negatively with adequacy (see Figs. 3.10 and 3.11), as might be expected, as a large distance indicates that many rare words have been added or (more likely) removed.

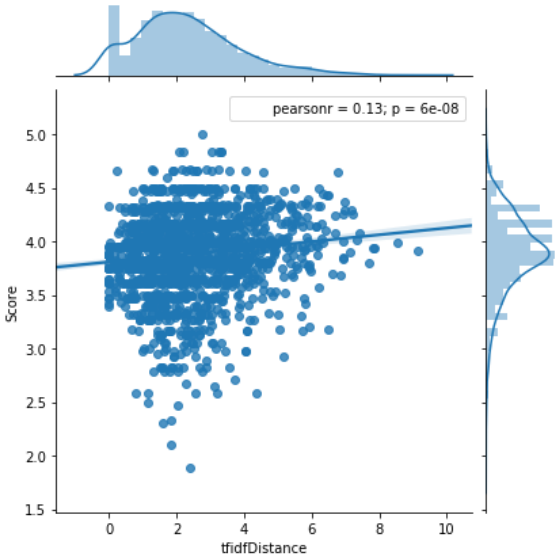

Figure 3.11: This plot demonstrates the importance of TFIDF as a metric. Note that it has a strong positive correlation with overall score (defined as the geometric mean of adequacy and simplicity), meaning that it is a useful feature in and of itself. Correlations calculated on training set.

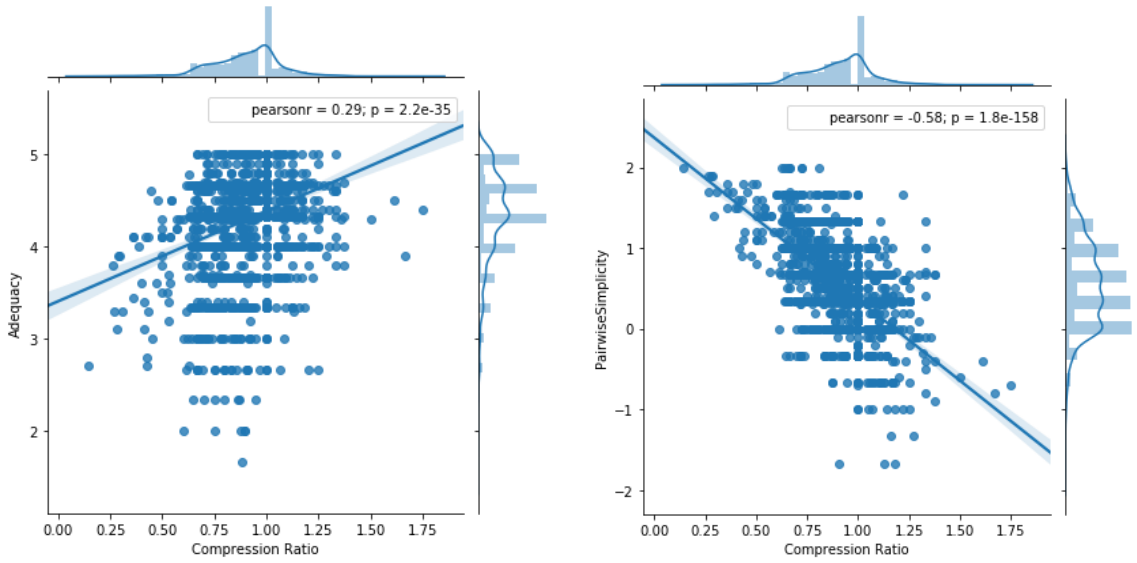

Figure 3.12: These plots analyze the importance of the compression ratio as a metric. Note that higher compression ratios indicate less length reduction. Compression ratio positively correlates with simplicity and negatively correlates with adequacy. Correlations calculated on training set.

3.5.5 Compression Ratio

The difference in length between a hypothesis and its corresponding original sentence was used as a feature. This feature, usually known as the compression ratio, was calculated as the ratio of total number of tokens in the hypothesis sentence to total number of tokens in the input sentence. In Figs. 3.12 and 3.13, we see that a lower compression ratio (indicating a greater reduction in length) has a strong positive association with simplicity, and a strong negative association with adequacy. All together, however, a low compression ratio is predictive of score, indicating that shorter simplifications are usually better than long ones.

3.5.6 System Label

We also included a feature indicating whether a sentence was an original human simplification or an output sentence from our sentence fusion system. In addition, each original sentence was also presented to the ranker with a special feature marking its identity, allowing the ranker to choose not to simplify a sentence in a case. This proved to be particularly valuable if adequacy

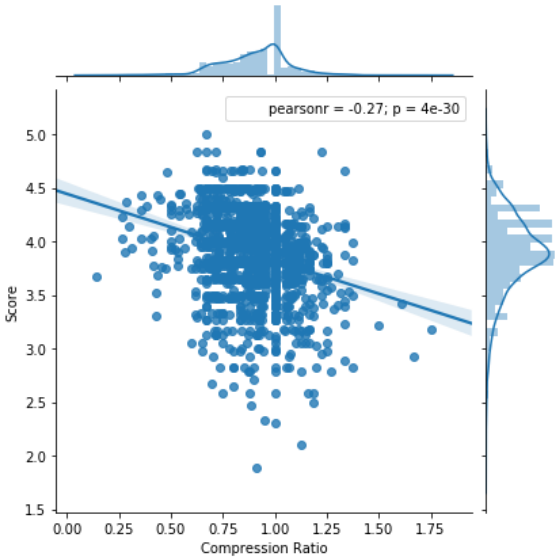

Figure 3.13: This plot demonstrates the importance of the compression ratio as a metric. Note that it has a strong positive correlation with overall score (defined as the geometric mean of adequacy and simplicity), meaning that it is a useful feature in and of itself. Correlations calculated on training set.

System	Accuracy Change
No sentence fusion data	-0.011
No Neural LM	-0.036
No Siamese	-0.003
No ngram LM	-0.012
No Compression Ratio	-0.099
No TFIDF distance	-0.006

Table 3.1: An ablation study demonstrating relative importance of our included features.

was prioritized. An analysis of how our ranker chose between human simplifications, sentence fusion output and original unsimplified sentences is presented in Fig 4.3 in Chapter 4.

3.5.7 Feature Choice

We conducted an ablation study to assess the importance of our reranking features, calculated for human simplifications only (Table 3.1). The baseline model had an accuracy of 0.703; note that this is a pairwise ranking task, so random chance produces an accuracy of 50%. In no case did excluding a feature improve accuracy, although not all features had a significant effect. A ranker trained *only* on human simplifications (line 2) is worse than a ranker trained on a combination of human simplifications and sentence fusion system output, so we use the ranker trained on both for all cases unless otherwise indicated. The substantial importance of our neural language model suggests that more advanced neural features might produce even better results.

3.5.8 Iterative Training

Our ranker requires a dataset consisting of multiple alternative simplifications of sentences. However, to our knowledge no such labeled training dataset exists. We were thus forced to create our own, using our own training data. One alternative available would have been to use the human simplifications available as a training set for the ranker. This, however, was undesirable, as the human simplifications differed greatly from the early output from our sentence fusion system. We thus discovered any ranker trained only with human simplifications would make extremely poor decisions when confronted with our system output.

To circumvent this issue, we needed to score sentence fusion output sentences in order to incorporate them into our training set. However, resource constraints made it impossible to score

all of the thousands of sentence fusion output sentences generated, even for a small number of input sentences. Thus, we sought to choose a representative subset of system output sentences to score. To do this, we employed a series of increasingly competent rankers to separate the system output sentences into deciles based on perceived quality, and then scored a highest-ranked example from each decile on Amazon Mechanical Turk (see Section 3.2). In this way, a dataset was created in an iterative manner.

The initial ranker used was one based only on the neural language model feature (see Section 3.5.1). As there was only a single input feature and our ranker is a linear-kernel SVM, the only information the ranker could learn was whether lower or higher language model probability was desirable. We decided that sentences judged by the language model to be more probable were, *a priori*, likely to be simpler, and set the ranker accordingly. We then generated deciles and had them evaluated on Amazon Mechanical Turk.

This data was then used to train a second ranker, in which we incorporated the full set of features used in our final model (see previous sections for details). This ranker was again used to separate the output system into deciles, which were evaluated. These results were then used to train a final ranker, for which the process was repeated. This ranker had approximately converged in quality with the original ranker, so we halted training after evaluating its results.

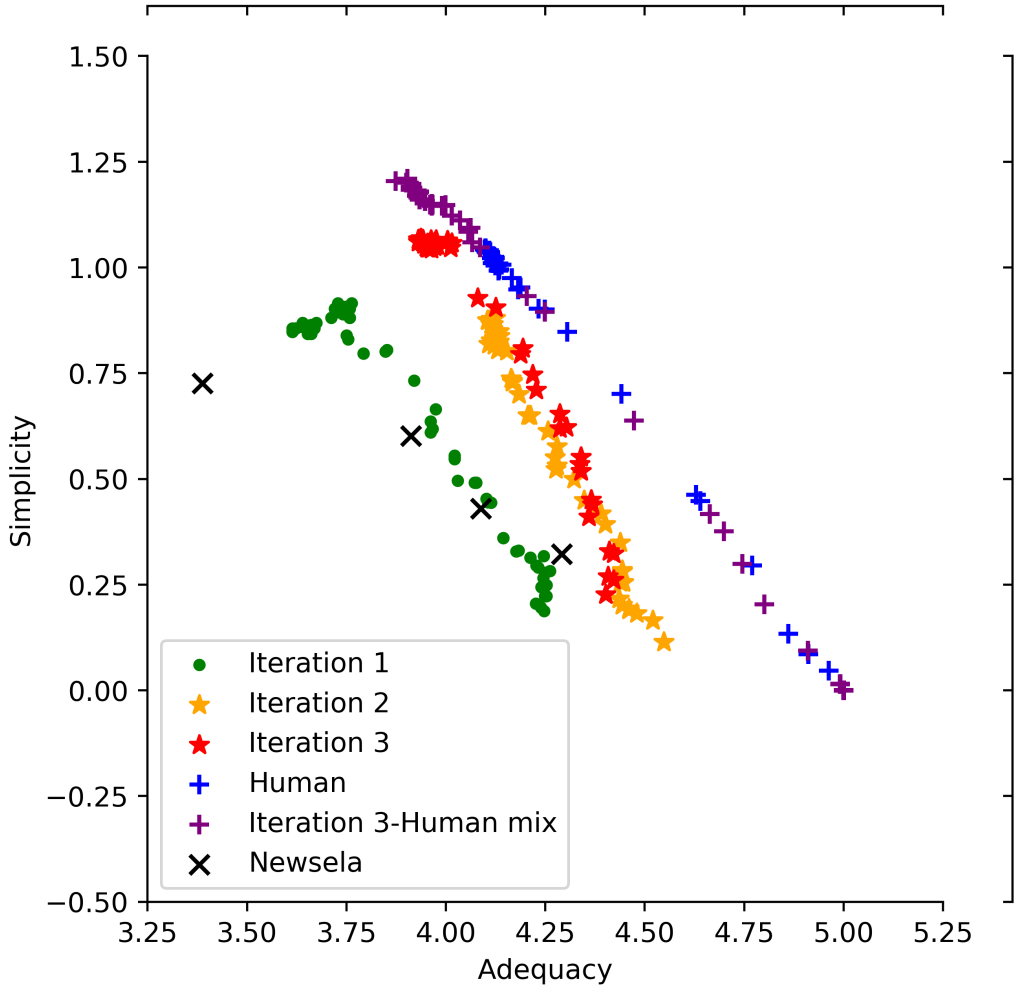

Figure 3.14: A figure showing progress in training across iterations.

Chapter 4

Results

We found that our system was convincingly able to beat our benchmark, Newsela, at every level of simplification. Fig. 4.1 shows a demonstration of this, using a wide range of adequacy/simplicity weightings. We find that our system outperforms each level of Newsela (V1, V2, V3 and V4) by a significant margin on both adequacy and simplicity at at least one adequacy/simplicity weighting point. Results for this are shown in Tables 4.1, 4.2 and 4.3.

We also examined the quality of our reranker. We first examined the difference between the best and worst candidate human simplifications for each input sentence (Fig. 4.2) according to our ranker, and found that there was a strong separation. This indicates that our ranker was reasonably learning to distinguish high- and low-quality simplifications.

We also analyzed what fraction of chosen sentences were synthetic (i.e., from the sentence fusion system), and what fraction were from amateur human simplifiers. Interestingly, this varied heavily across different relative weightings of adequacy and simplicity, with higher priority placed on simplicity leading to more synthetic sentences being chosen, up to a maximum of

System	Simplicity	Adequacy
Joint System	**0.81**	**4.33**
Newsela V1	0.61^{**}	4.11^{***}
Newsela V2	0.65^{*}	3.96^{***}

Table 4.1: This table shows a comparison of our system to Newsela V1 and V2, the least-simplified two versions of the Newsela corpus, over our testing set. The version of the system shown is that with an adequacy/simplicity weighting of $\frac{5}{3}$. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

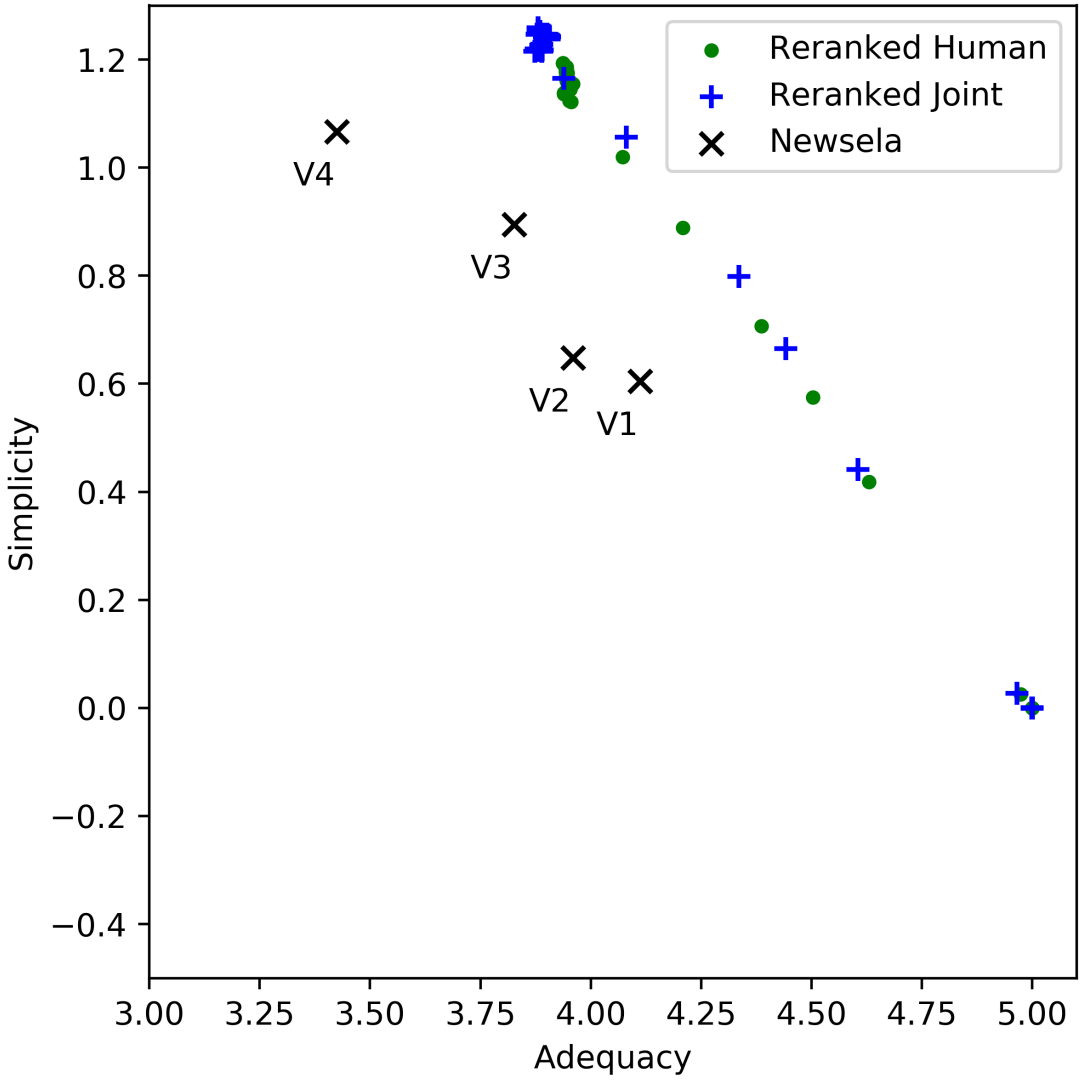

Figure 4.1: A comparison of output from our system vs the benchmark Newsela sentences, across a wide range of relative adequacy/simplicity weightings. Error bars are not included, for visual clarity; significances of key comparisons are provided in Tables 4.1 and 4.3.

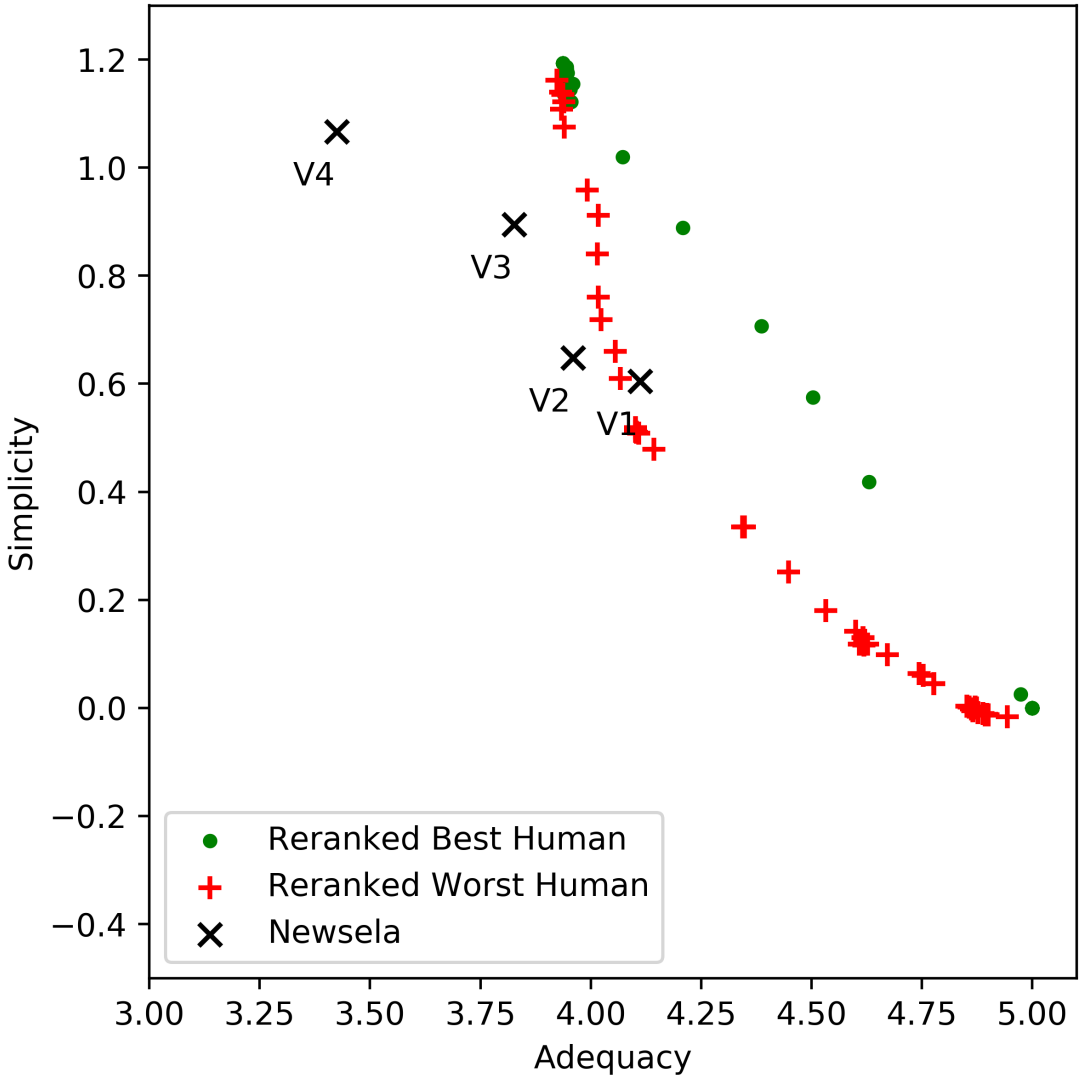

Figure 4.2: An illustration of our ranker's ability to differentiate between low- and high-quality simplifications. Performed only for human simplifications, as the worst sentence fusion system output would be too poor to be an interesting comparison.

System	Simplicity	Adequacy
Joint System	**1.06**	**4.08**
Newsela V3	0.90^{**}	3.83^{***}

Table 4.2: This table shows a comparison of our system, with and without sentence fusion sentences, to Newsela V3, the second-most-simplified version of the Newsela corpus, over our testing set. The version of the system shown is that where the adequacy/simplicity weighting is 1.5625. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

System	Simplicity	Adequacy
Joint System	**1.26**	3.88
Human-Only	1.19^{*}	**3.94**
Newsela V4	1.06^{**}	3.43^{**}

Table 4.3: This table shows a comparison of our system, with and without sentence fusion sentences, to Newsela V4, the least-simplified version of the Newsela corpus, over our testing set. The versions of the system shown are those where the adequacy/simplicity weighting is 0/1. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

roughly 40%. In addition, we examine what fraction of chosen sentences are in fact the original, *unsimplified* sentences; unsuprisingly, this goes to 1 if adequacy is given absolute priority, as unsimplified sentences by definition have perfect adequacy. The results can be seen in Fig 4.3.

To better assess the quality of our ranker, we also considered the use of an oracle, where we directly use human annotations for sentence selection. This is in fact analogous to the approach taken by Zaidan and Callison-Burch [2011], and represents a practical approach when the extra expense is permissible; choosing to use human annotators instead of a ranker would result in roughly a doubling of costs on Amazon Mechanical Turk. We first find in Fig. 4.4 that the use of such an oracle provides large quality increases over our reranker, when only human simplifications are taken into account. Intriguingly, we note that for a certain adequacy/simplicity weighting, our oracle method produces simplifications which are both significantly more adequate than Newsela V1 and significantly more simple than Newsela V4, *at the same time*, as can be seen in Table 4.4.

We also examined the inclusion of sentence fusion sentences chosen by the ranker in the oracle study. This is analogous to using a two-stage system, where our reranker is allowed to find cases where it predicts that a synthetic simplification will be superior to the available human simplifications, and adding all such synthetic simplifications to the set of human simplifica-

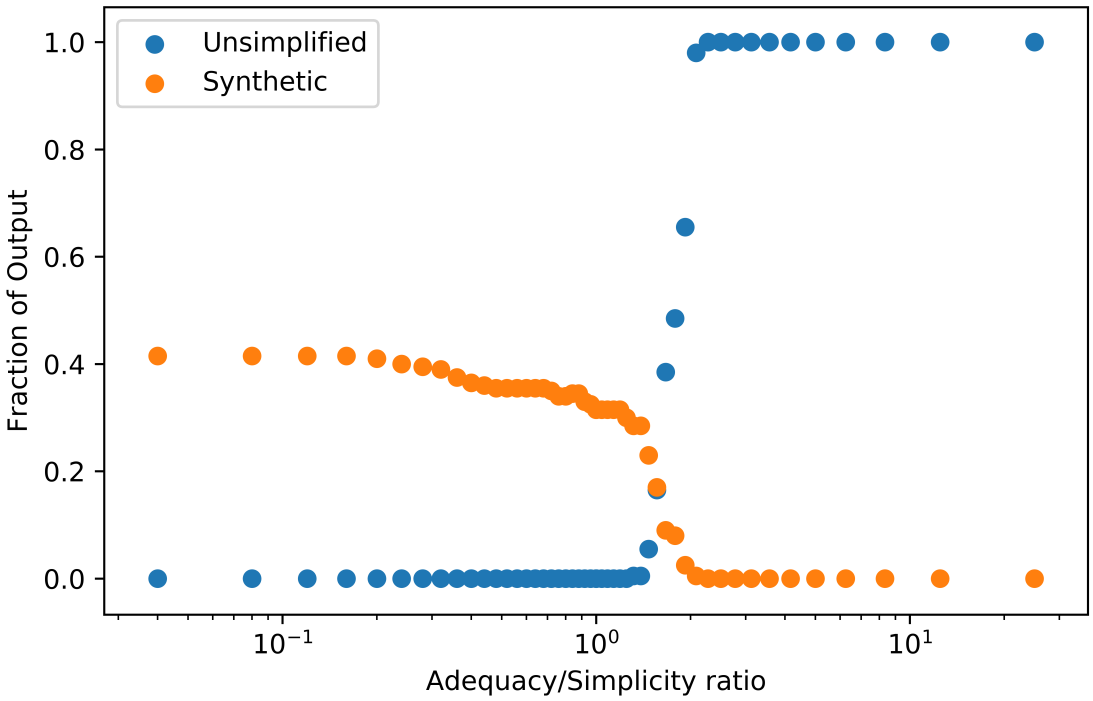

Figure 4.3: The fractions of output sentences coming from the sentence fusion system and output sentences which are unsimplified, shown against the relative weightings of adequacy and simplicity. Note that when only simplicity is prioritized roughly 40% of sentences are from the sentence fusion system, illustrating that the sentence fusion system produces simple output. Note also that high preferences for adequacy versus simplicity lead to only unsimplified sentences being chosen.

System	Simplicity	Adequacy
Oracle	**1.23**	**4.31**
Newsela V1	0.61***	4.11***
Newsela V4	1.06**	3.43***

Table 4.4: This table shows a comparison of our human simplifications, with an oracle applied, to Newsela V1 and V4. We show that, with an oracle, we are able to simultaneously outperform Newsela V1, the least-simplified version, on adequacy, while outperforming Newsela V4, the most simplified version, on simplicity. The oracle shown was performed with an adequacy/simplicity weighting of 1/1. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

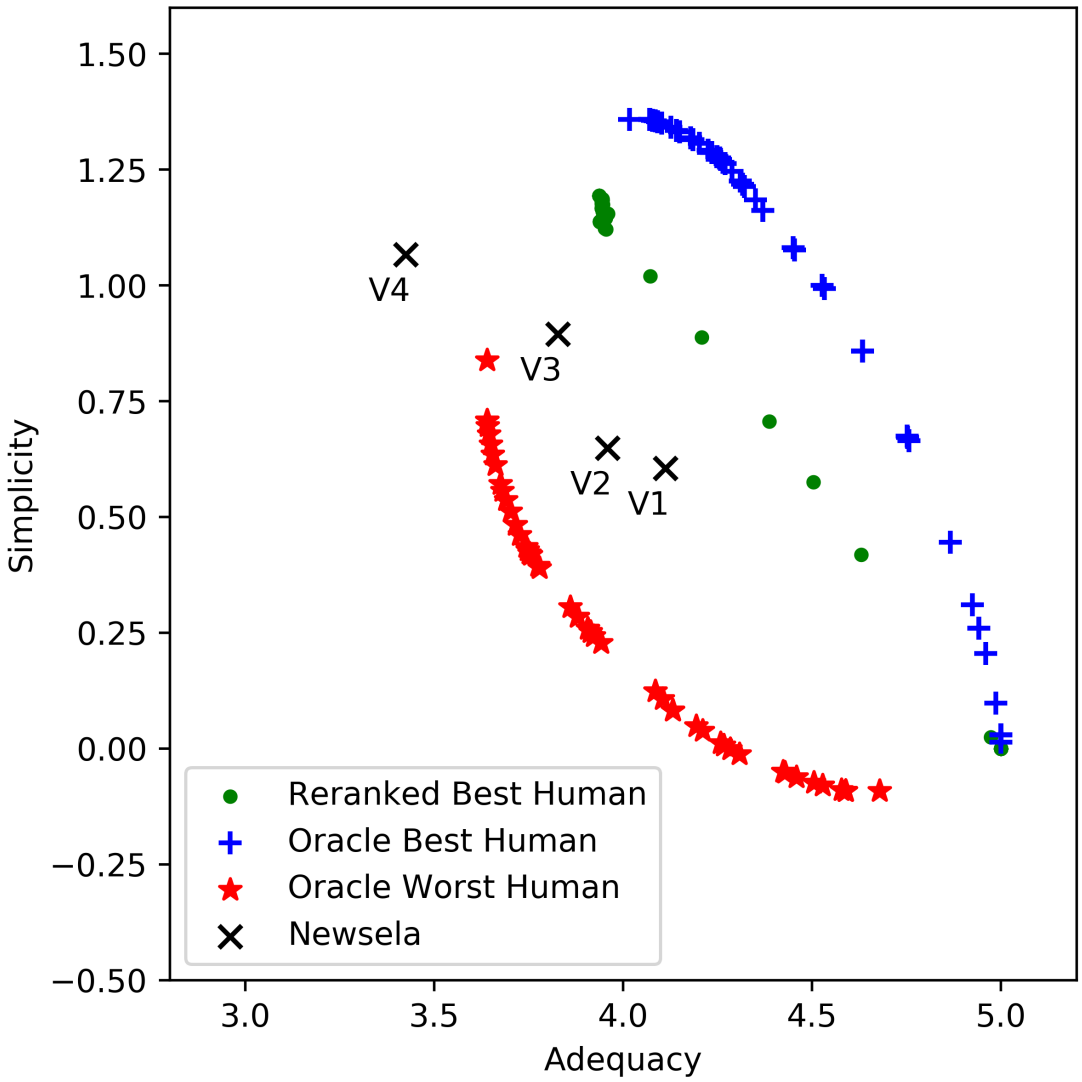

Figure 4.4: An illustration of the outer bounds on ranker performance, performed by oracle (i.e., choosing best and worst sentences for combinations of adequacy and simplicity). Note that this was performed only for human sentences, as annotating all synthetic sentences would be prohibitively expensive. It can be seen that some room for improvement remains, although our ranker is clearly quite capable.

System	Simplicity	Adequacy
Joint Oracle	**1.31**	**4.33**
Human-only Oracle	1.23^{***}	4.31
Newsela V1	0.61^{***}	4.11^{***}
Newsela V4	1.06^{**}	3.43^{***}

Table 4.5: This table shows a comparison of our human simplifications, with an oracle applied, to Newsela V1 and V4. We show that, with an oracle, we are able to simultaneously outperform Newsela V1, the least-simplified version, on adequacy, while outperforming Newsela V4, the most simplified version, on simplicity. The oracles shown were performed with an adequacy/simplicity weighting of 1/1. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

tions to be evaluated on by human annotators. This yields a significant increase in performance on simplicity over only using human simplifications, as can be seen in Fig. 4.5 and Table 4.5. Intriguingly, we found that the oracle system was even more consistent than our reranker in its use of synthetic sentences; an analysis can be seen in Fig. 4.6.

Finally, we conducted an experiment where we *also* applied an oracle to Newsela, allowing the oracle to choose between Newsela V1 through V4 simplifications just as it would choose between human simplifications (Fig 4.7). Intriguingly, we found that including the synthetic sentences suggesting by our ranker allowed us to maintain a complete superiority over Newsela, even with the oracle applied, *especially* when the our weighting was geared to emphasize simplicity over adequacy. Without the added synthetic sentences, in fact, our human simplifications were in most cases little better than Newsela – at none of the adequacy/simplicity weightings tested did our human-only system ever beat the corresponding Newsela oracle on both adequacy and simplicity by a statistically significant margin ($p < 0.05$ – no multiple-comparisons correction was used, since significance was never achieved). With synthetic sentences added to the mix, however, our system was able to do so, as can be seen in Table 4.6.

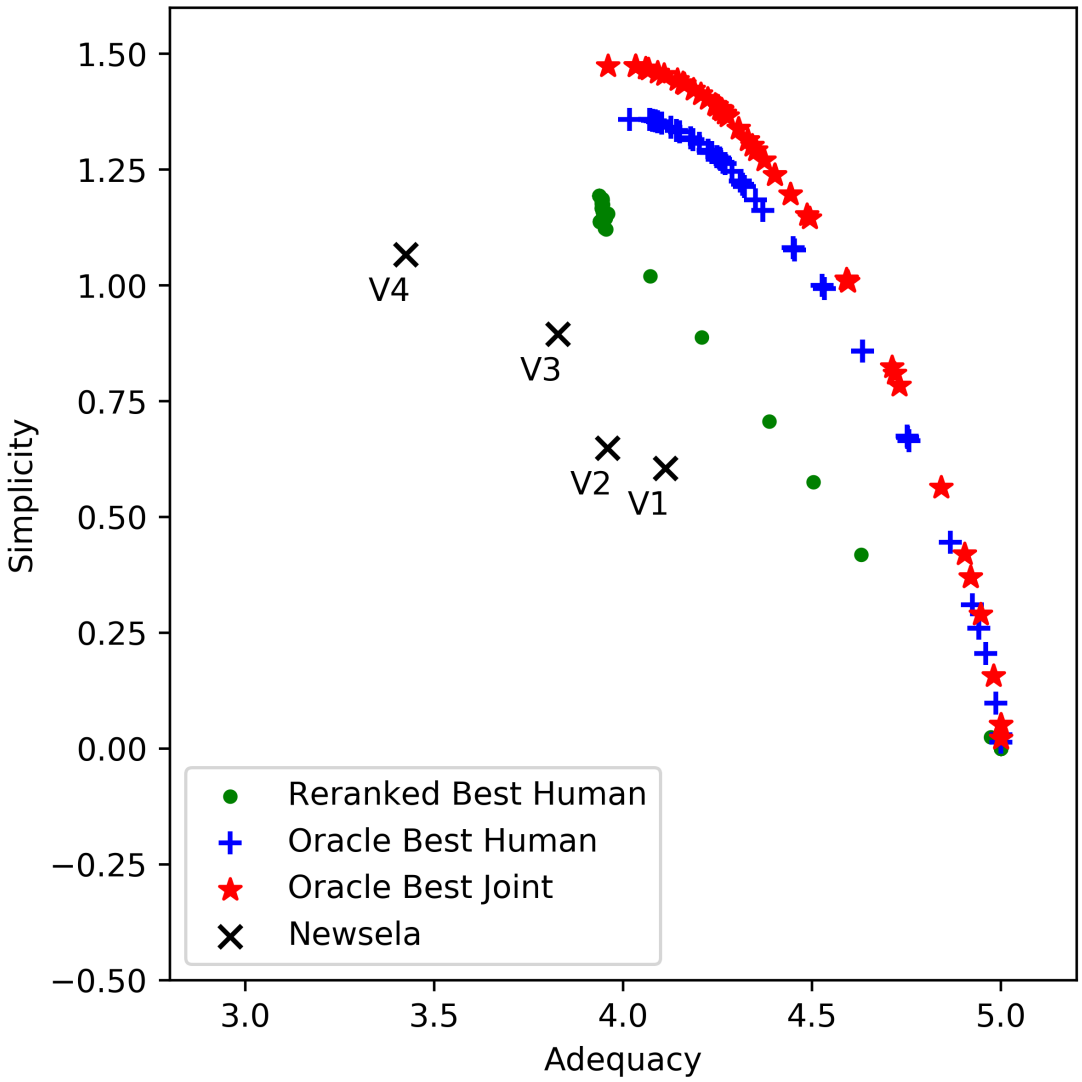

Figure 4.5: An illustration of the outer bounds on ranker performance, performed by oracle (i.e., choosing best and worst sentences for combinations of adequacy and simplicity). This was performed for human simplifications and synthetic simplifications flagged by the ranker. It can be seen that including synthetic simplifications significantly increases the maximum simplicity achievable, at no cost to adequacy.

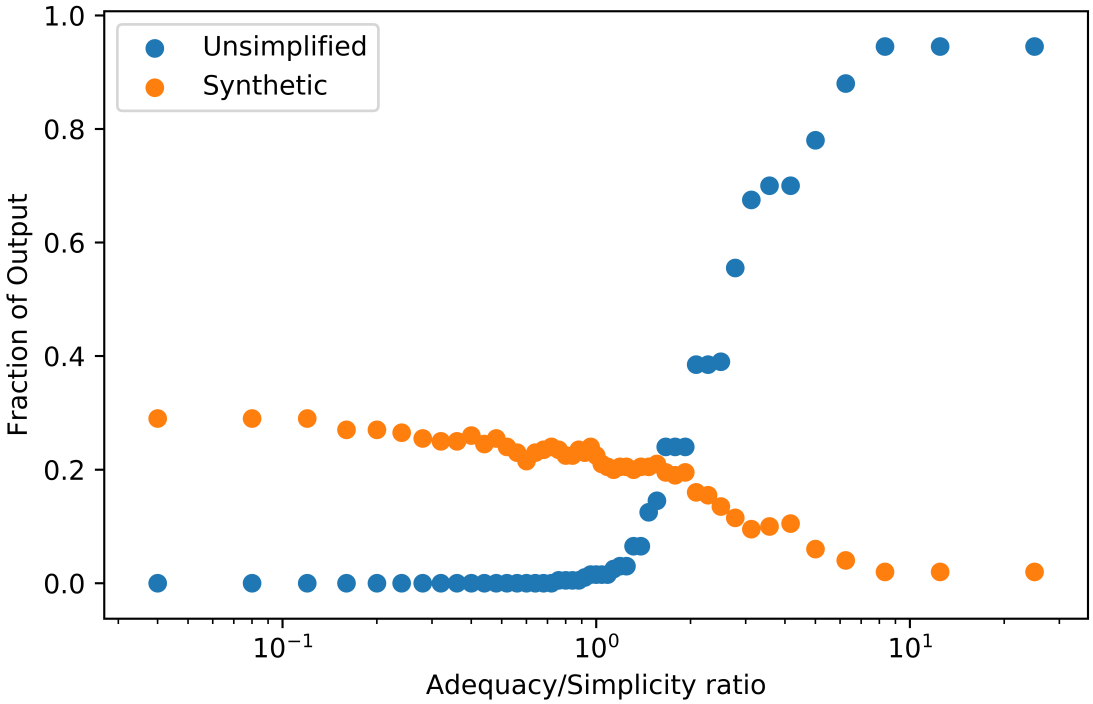

Figure 4.6: The fractions of output sentences coming from the sentence fusion system and output sentences which are unsimplified as chosen by oracle, shown against the relative weightings of adequacy and simplicity. Note that when only simplicity is prioritized roughly 30% of sentences are from the sentence fusion system, down from the fraction of roughly 40% chosen by our reranker. However, the oracle does consistently choose to use sentences from the sentence fusion system when simplicity is prioritized, again illustrating that the sentence fusion system produces simple output. Note also that, as with our ranker, high preferences for adequacy versus simplicity lead to only unsimplified sentences being chosen.

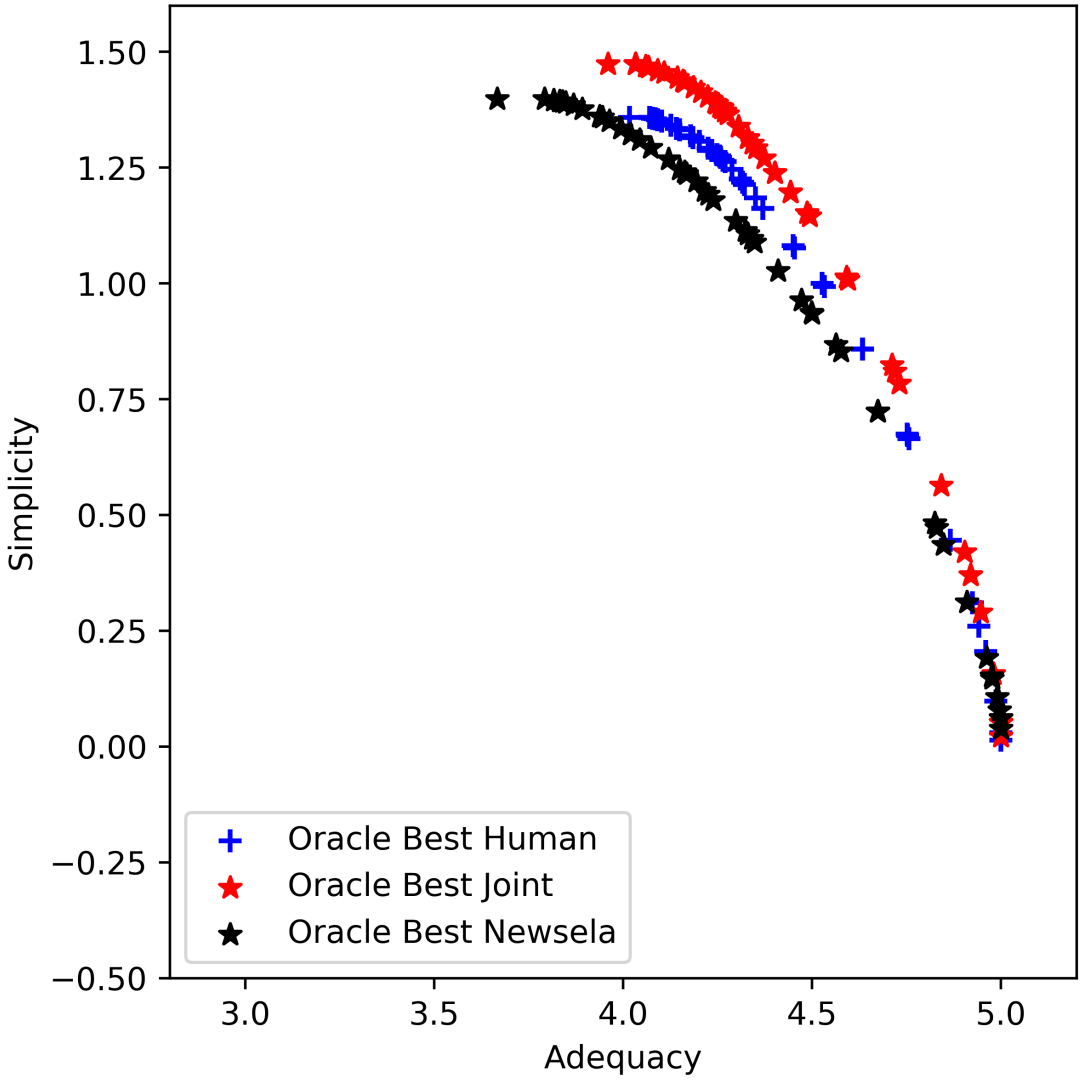

Figure 4.7: A comparison of our system, with an oracle and with and without synthetic sentence added, to an oracled version of Newsela. Note that the synthetic sentences (joint system) are vital for maintaining our advantage over Newsela.

System	Simplicity	Adequacy
Joint Oracle	**1.39**	**4.25**
Human-only Oracle	1.29***	4.24
Newsela Oracle	1.27**	4.12**

Table 4.6: This table shows a comparison of our system, with an oracle applied with and without sentence fusion sentences, to Newsela with an oracle applied. We show that, with the added synthetic We show that, with sentence fusion sentences added (joint system) we are able to outperform Newsela on both adequacy and simplicity by a significant margin, while also beating a version of our system which includes only crowdsourced human simplifications. The oracles shown were performed with an adequacy/simplicity weighting of 3/5. * denotes $p < 0.05$, while ** denotes $p < 0.01$ and *** denotes $p < 0.001$.

Chapter 5

Discussion

Given the ease with which we were able to outperform Newsela, a fundamental investigation into the weakness of the Newsela dataset would be appropriate. Informally, it appears that the sentence alignments provided by Xu et al. [2015] may be imperfect, leading to Newsela simplifications having artificially low adequacy due to sentence splittings occurred during simplification but are not respected in the alignments. We sought to counteract this by grouping split Newsela simplifications together (see Section 3.1), but this may not have been completely successful.

Nevertheless, we argue that the work here represents a solid foundation for text simplification crowdsourcing research. The largest avenues for future research relate to the reranking and sentence fusion schemes in place. As is clear from the oracle study in Fig. 4.4, there is much room remaining for improvement in our reranking scheme. Future research could examine the incorporation of new features (such as a custom-trained neural network feature based off of our Siamese LSTM model [Mueller and Thyagarajan, 2016], or a convolutional model, as in the work of Severyn and Moschitti [2015]). Additionally, it might be possible to use more capable neural language models as a foundation for other features, as was done by Radford et al. [2017] in predicting sentiment. Outside of this, the implementation of more sophisticated reranking schemes could also have a large impact; in particular, more sophisticated list-based ranking systems [Cao et al., 2007] have shown good performance, and might be appropriate for this task.

Additionally, experiments could be conducted on the use of multiple synthetic sentences in an oracle-style human evaluation sentence selection system. In this work, we only included a synthetic sentence when it was ranked more highly than all other sentences by our ranker. However, there is no reason besides cost not to evaluate even more synthetic sentences, as doing

so could lead to some higher-quality sentences being found. This is limited only by the number of annotations available, and is thus primarily an experimental concern.

Moreover, there is also room for the use of entirely different sentence fusion systems. In this work, we examined only the model proposed by Filippova [2010], albeit with the alignment modifications described in Section 3.4.1. That said, there is no *a priori* reason to hold that other sentence fusion systems could not produce superior output. In particular, with the recent development of neural systems for abstractive text summarization [Chopra et al., 2016; Nallapati et al., 2016], a neural approach to sentence fusion in this context seems increasingly viable. The primary concern is that such a neural model would require the existence of a supervised dataset, but some other dataset might be adapted to suit this. Alternatively, it might be possible to synthetically create such a dataset from the Newsela corpus; for example, one could train an abstractive neural text summarization system to predict Newsela V4 from collections of aligned Newsela unsimplified, V1, V2 and V3 sentences.

Bibliography

Marcelo Amancio and Lucia Specia. An analysis of crowdsourced text simplifications. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

B Bangalore, German Bordel, and Giuseppe Riccardi. Computing consensus translation from multiple machine translation systems. In *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*, pages 351–354. IEEE, 2001.

Regina Barzilay and Lillian Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 16–23. Association for Computational Linguistics, 2003.

Regina Barzilay and Kathleen R McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.

Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 286–295. Association for Computational Linguistics, 2009.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013. URL `http://arxiv.org/abs/1312.3005`.

Jackie Chi Kit Cheung and Gerald Penn. Unsupervised sentence enhancement for automatic summarization. In *EMNLP*, pages 775–786, 2014.

François Chollet et al. Keras. `https://keras.io`, 2015.

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

Trevor Cohn and Mirella Lapata. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 728, 2007.

William Coster and David Kauchak. Learning to simplify sentences using wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9. Association for Computational Linguistics, 2011.

Hal Daume III and Daniel Marcu. Generic sentence fusion is an ill-defined summarization task. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2004.

Katja Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics, 2010.

Katja Filippova and Michael Strube. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. Association for Computational Linguistics, 2008.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*, 2016a.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. Zero-resource translation with multi-lingual neural machine translation. *arXiv preprint arXiv:1606.04164*, 2016b.

Robert Frederking and Sergei Nirenburg. Three heads are better than one. In *Proceedings of the fourth conference on Applied natural language processing*, pages 95–100. Association for Computational Linguistics, 1994.

Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268, 2010.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. 1999.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*, 2016.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

Emiel Krahmer, Erwin Marsi, and Paul van Pelt. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 193–196. Association for Computational Linguistics, 2008.

Walter S Lasecki, Luz Rello, and Jeffrey P Bigham. Measuring text simplification with the crowd. In *Proceedings of the 12th Web for All Conference*, page 4. ACM, 2015.

Ching-Pei Lee and Chih-Jen Lin. Large-scale linear ranksvm. *Neural computation*, 26(4):781–817, 2014.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics, 2006.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics, 2002.

Wei-Yun Ma and Kathleen McKeown. System combination for machine translation through paraphrasing. In *EMNLP*, pages 1053–1058, 2015.

Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Fabian Pedregosa, Elodie Cauvet, Gaël Varoquaux, Christophe Pallier, Bertrand Thirion, and Alexandre Gramfort. Learning to rank from medical imaging data. In *International Workshop on Machine Learning in Medical Imaging*, pages 234–241. Springer, 2012.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, 2007a.

Antti-Veikko I Rosti, Spyridon Matsoukas, and Richard Schwartz. Improved word-level system combination for machine translation. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 312, 2007b.

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM, 2015.

Advaith Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014.

Khe Chai Sim, William J Byrne, Mark JF Gales, Hichem Sahbi, and Philip C Woodland. Consensus network decoding for statistical machine translation system combination. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–105. IEEE, 2007.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

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

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.

Omar F Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1220–1229. Association for Computational Linguistics, 2011.

Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*, 2017.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. Neural system combination for machine translation. *arXiv preprint arXiv:1704.06393*, 2017.

Barret Zoph and Kevin Knight. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*, 2016.