Simplifying English Texts Using Statistical Machine Translation

Alexandru Grigoras
NYU Shanghai
ag4601@nyu.edu

Marcus S. Arellano
NYU Shanghai
msa455@nyu.edu

Abstract

Text simplification is defined as the process of reducing the syntactic and grammatical complexity of a text, while preserving the meaning and information contained in the original version. Manual simplification has been proven to be highly effective, and has been used in a number of fields and communities, the most popular perhaps being the "Simple Wikipedia". Automatic text simplification, however, is a relatively new research field, and has been approached in a multitude of ways. Our paper aims to explain our own approach to the problem: with a predefined vocabulary and an approximately parallel corpus, we implemented IBM Models 1 and 2 and attempted to perform text simplification through statistical translation. The goal of this paper is to present an innovative way to address this problem and, hopefully, to push this research forward in a different direction.

1 Introduction

The main purpose of text simplification is to make legal, philosophical or technical texts more accessible to people who do not possess the necessary, highly specialized vocabulary (majority of speakers of any language), while also enabling beginners to more easily dive into a new culture and be able to interact with a variety of materials written in the new language. Moreover, text simplifcation can have much more far-ranging applications. It could bring significant improvements to other Natural Language Processing tools such as summarization or machine translation, and even be used to help the learning process of people with certain disabilities.

One of the factors that makes text simplification extremely difficult is the lack of existing parallel corpora. Furthermore, there is yet no general agreement upon what constitutes simplified text (acceptable vocabulary, grammar, syntax, etc.), people in different fields often using their own simplified version of English (such as the Simplified Technical English used in the Aerospace industry). Not even the Simple Wikipedia can be considered to be a more universal standard, since both linguists and the general public often disagree over whether it is actually "simple" enough.

For this reason, we have decided to begin this project with a clear set of rules. After extensive research, we chose Charles Kay Ogden's "Basic English" as our desired target. Ogden is a linguist who dedicated most of his work to developing a highly simplified version of English that can convey all of the necessary information for a person with average English skills. This subset of English is limited to only 850 base words which, together with no more than 150 slightly more specialized words, should be enough to reproduce most English texts.

In addition to designing "Basic English" and its rules, Ogden also manually simplified a number of texts in order to showcase the strengths of his method. Unfortunately, his translations were not done directly in a word-for-word (or even sentence-for-sentence) manner, which made us initially dismiss the possibility of using these existing texts as training corpora. However, one of the books that Ogden rewrote in full is the Bible. While due to its religious and very old vocabulary, the Bible definitely does not make an ideal candidate, we decided to still use it as our parallel corpus thanks to one of its unique features: verse numbering. We realized that used together with the standard version of the Bible, we could index every verse and thus create a parallel corpus of

perfectly aligned verses. Some of the downsides include the fact that verses can sometimes be very long, and that they are not necessarily the same as sentences, which would lead to a slightly lower accuracy than could be obtained with a specially designed simplification corpus.

2 Data Collection

Our parallel corpus was designed to consist of two main texts: a) the original King James Bible (referred to as the "Regular English Corpus") and b) Ogden's translation of the Bible (also named the "Basic English Corpus").

For our Regular English Corpus, we used the publicly available version offered by Project Gutenberg, without any modification.

For the Basic English Corpus, we developed a personalized web scraper to collect the data from `basic-english.org`. The scraper initially parses the list of books of the bible and compiles a list of book names, and a list of links. However, each book of the Bible is divided into chapters which are stored at different locations on the website, so the scraper looks for all of the unique links and groups them by book, in order to ensure consistency. Then, for each chapter of each book, it navigates to the corresponding web address, and collects all of the text. This resulting material is then compiled into one file, constituting the entire Bible written in Basic English.

3 Data Extraction and Indexing

3.1 Data Extraction

The verse numbering property of the bible might initially make the task of extracting verse data seem trivial, yet this has hardly been the case. While the Regular English version of the Bible had rather consistent formatting which made extraction quite straightforward, the Basic English version was overwhelmingly inconsistent, particularly in terms of verse numbering. A very large number of verse numbers were incomplete, while some were fully missing. An even greater challenge was the fact that there was a mistake in the book order, and an entire chapter left untranslated, difficulties leading to unreconcilable differences between the corpora.

We did, however, identify all of these issues in due time and revised both our scraping algorithm and our data extraction one in order to capture these mistakes and fix them in the best possible way. We manually examined the verse numbering inconsistencies and devised a very effective way to infer the chapter and/or verse numbers for the cases where they were missing, accounted for any missing or incomplete data, and then generated a 99% accurate representation of the Bible, in both Basic and Regular English.

3.2 Indexing

Once the data was successfully extracted, we made a full verse index for each version of the Bible. We tokenized the sentences and cleaned up the formatting, then stored all of the data in a Python dictionary addressable via the following syntax: **Bible[BookName][ChapterNumber][VerseNumber]**. In addition to being particularly useful for us in generating the simplifcation parallel corpus, this dataset could potentially be very useful in any number of other projects which may want to use the Bible as a dataset, and specifically any further research into Ogden's Basic English. We are thus considering releasing this indexed parallel corpus as a public dataset.

4 The Translation Model

4.1 General Principles

Having already compiled a parallel dataset, the next step in the translation process is developing a statistical translation model.

The basic idea behind the process is that each word/phrase from the source "language" (in our case, Regular English) has a direct equivalent in the target "language" (Basic English). The task of the program would be to generate a model that can automatically infer these equivalencies.

The modeling process begins with an initial model which assumes that any Regular English word could be "translated" to any Basic English word with equal probability. To increase efficiency, we only consider real the possibilities defined on words which belong to equivalent/parallel sentences in our corpus. The algorithm then iterates over the datset for a number of times (ideally until it converges), and with each iteration it adjusts the probabilities based on the observations made in the dataset.

Here is a quick example of how the algorithm works. Let's assume an initial parallel corpus of only three short phrases (per language).

	Regular English	Basic English
1	abrogate legislation	revoke law
2	arbitrary legislation	random law
3	arbitrary decision	random choice

Let's take the word "legislation" as our example. When the model is initialized, it is equally likely to be translated as "revoke", "law", or "random" (since these are all of the Basic English words appearing in the sentences that correspond to sentences containing the word "legislation"). After the first iteration, however, there is a 50% chance that its equivalent is "law", and 25% each for "revoke" and "random" (since "law" appears twice). Thus, even if analyzed alone, the best translation for the word "legislation" would be "law" - which is correct. However, the real strength of the model lies in the fact that the probabilities of all words, not just one, get updated with every single iteration. This means that, in our case, the word "arbitrary" will also (independently) have a 50% chance to be translated as "random". When put together, these two observations actually increase the odds of the correct translation of both terms, since each of them becomes less likely to be translated to the same word as the other. As a result, the model would converge with the following translation probability table:

Regular English	Basic English	Probability
abrogate	revoke	1.0
arbitrary	random	1.0
decision	choice	1.0
legislation	law	1.0

While this explanation is oversimplified, and real corpora are far more complex, the general principles remain the same and have historically led to admirable machine translation tools (although they may no longer be able to directly compete with state-of-the-art algorithms).

4.2 IBM Model 1

The IBM alignment models are a series of models such as the one briefly described above, which are commonly recognized and used in a multitude of research projects.

Model 1 is the first of these models and the least complex of all. It is focused on lexical translation and generates a very basic translation model (which, however, lies at the base of all of the following ones). The algorithm behind Model 1 works almost exactly as described above, and generates a probability matrix which shows the best word-for-word translations, without having any concept of ordering.

4.3 IBM Model 2

IBM Model 2 is the next iteration of the same model, which relies on the same principles of Model 1, but also introduces an additional absolute alignment model.

The driving need for Model 2 was the fact that Model 1 considered any ordering of a sentence as equally likely, which meant that it essentially treated the sentence as a bag of words. With the addition of the ordering concept, Model 2 solves the issues caused by Model 1 and leads to improved and more contextually-aware translation, which also opens up the way for phrase-based translation.

4.4 Other Models

IBM Models 3-6 introduce a number of other features, including a fertility model, a relative alignment model, and HMM. All of these features would considerably improve the translation quality of our algorithm, yet the scope of this project was limited to the first two IBM Models.

5 Sentiment Analysis

5.1 Verification

When initially planning this text simplification project, one of the difficult pieces was a verification tool. As the program takes a complex text as input and returns a simple text as output, checking the accuracy of the translation is just as important as making sure the words used are simpler than those in the original text.

Traditionally, this task would have to be done by an actual human verifying the accuracy of the translations one by one. However, in recent years there have been major advancements in the field of semantic analysis, or extracting actual meaning from text. Many companies have developed methods of classifying text depending on the feelings behind it. For example, Microsoft has recently released a Cognitive Services API with many capabilities, such as scoring text on a percentage scale of positivity, creating language understanding models to execute operations, and even some chat bot toolkits to synthesize natural conversation and respond to actual user queries.

We decided to look into this field to try and develop a method of scoring our two texts and verify

that their meanings were similar. As our translation tools would only create output texts with words from the simplified corpus, we can be sure our texts were simplified. Being able to check they have the same meaning would ensure that the translation was a success.

5.2 Semantic Similarity

For the task of comparing the texts' meanings, we looked into a specific field of sentiment analysis: semantic similarity. In many cases, this is used on entire documents, for example when comparing TF-IDF scores of various documents to find answers relevant to a user query. However, our goal wasn not to simply paraphrase the entire document, but instead we wanted to analyze and compare the content sentence by sentence. This way, our comparison is much more granular and can more easily identify translation errors.

6 Similarity Model

6.1 Ideal Model

What we aimed to build was a tool that would be able to process the original document and the simplified document created by the simplification tool. It would then convert the documents into a list of sentences, with each sentence being a list of words respectively. Using a corpus relevant to the subject matter of the documents, we trained a word embedding model to predict the context words around a given word. By comparing common context words of two different terms, we can effectively compare their semantic similarity, as proven by Tomas Mikolov.

After training our model, we then iterated through our two lists containing the sentences from our two documents and converted them into vectors. We used the resulting model to compare the sentences to each other one at a time. Finally we output a list of weights, each corresponding to the respective sentences that were compared. Each weight will be a value between 0 and 1 comparing the similarity of the sentences.

6.2 Word2Vec

Word2vec is a package of neural networks used to construct contexts of words. As deep learning has become a larger point of interest for many researchers attempting to solve long standing NLP problems, word2vec gained attention by demonstrating how "shallow learning" can outperform many new deep learning approaches.

The neural networks in this package are only two layers deep, and aim to leverage the power of a large and verbose starting dataset over that of an intense algorithm. The algorithm itself works by looking at the words surrounding a target word and charting the frequency of that word appearing in specific contexts. After being trained on enough examples of actual sentences, the model can then predict the word given the context around it, or predict the reverse, a context given an individual word. This context generation can effectively be used to compare the semantic similarity of various terms or phrases, as similar words appear around similar other sets of words.

Once vectorized, we can actually work out relations between different words in a similar way to how analogies work. For example, "king" is to "queen" as "man" is to "woman". Taking the vectors of "king", "queen", "man", and "woman", the equation $v_{queen} = v_{king} - v_{man} + v_{woman}$ will hold true. As the vectors behave this way, we can directly compute similarity of words as well as similarity of phrases. The gensim library for python makes utilizing word2vec simpler so we will be using it for the comparison modules.

7 Project Modules

The code behind this entire project is divided into a number of separate modules.

7.1 Scraper

The "scraper" module collects information from Ogden's website and automatically compiles the Basic English Bible.

7.2 Bible Extractor

The "extractor" has methods for parsing both the Regular English and the Basic English Bibles. It extracts the entire contents of the Bible given and divides it into books, chapters, and verses, returning a fully indexed version of the data.

7.3 Model Generator

The "model generator" has our own implementations of IBM Models 1 and 2 and has methods that generate the probability and alignment models given any parallel corpus.

7.4 Trainer

The "trainer" module is a comprehensive interface for the Model Generator. It enables customization

of name and number of iterations, as well as the automatic extraction of Bible text or the use of a different corpus instead, and the ability to choose between our custom implementation or NLTK's official implementation of IBM Model 2.

7.5 Translator

The "translator" module contains methods for quick translation using either Model 1 or Model 2 data. It can perform two types of translations: 1) word-by-word, using only a probability table; 2) aligned word-by-word, using a probability table and an alignment model. We also had a beta implementation of a phrase-based translation, yet the results obtained with IBM Model 2 were modest.

7.6 Sentiment Analysis Data Processing

This second preprocessing module works with the two documents from the translation output, the first being the original text, and the second being the simplified text. As these texts were already cleaned and preprocessed in the previous modules, we have no major scraping or cleaning tasks for these files. All that we must do to the text is remove stop words and punctuation, and stem (or lemmatize) the individual words as final processing.

7.7 Word2Vec Corpus Selection

Once the text has been processed, we can then use the gensim library for Python to implement word2vec. The only action needed for this step is choosing a corpus to train our model on.

The main strategy of word2vec is prioritizing size and depth of a corpus over power of your chosen algorithm. For this reason, we needed an extremely large and extensive set of text for our model to train on. We looked into previous experiments with word2vec and what sets of text other people have chosen to work with. Many people have had success using the entire repository of articles on Wikipedia, as this provides a large and verbose set of terms encompassing many different topics. We decided to use this repository to train our model as well.

However, due to the nature of word2vec, our model is only as good as the balance between our training and our testing corpora, which means that although this implementation would have good results on most texts, it proved to be unfit for our Bible corpus. Thus, in our final version, we train the word2vec model on the Bible instead.

7.8 Model Training and Comparison

We tested training the model both with the skip gram with negative sampling (SGNS) method as well as the continuous bag of words (CBOW) method and SGNS ended up creating a more reliable model for similarity comparisons.

After creating our model, we used gensim to convert the lists of sentences from the two documents into lists of vectors. We then iterated through the lists to compare sentence vectors to each other based on the word embeddings model. Finally we received lists of weights as an output to gauge the similarity of the sentences from both documents.

We then read through the simplified texts and compared them to the original texts to manually check similarity and confirm the similarities given were accurate.

8 Results

The scraping process was entirely successful, managing to compile the full Basic English Bible.

The extraction and indexing processes created significant difficulties, yet all of them were eventually solved through a combination of regular expressions and manual observations.

Model generation was relatively easy for IBM Model 1, but much more difficult to implement for Model 2. The multidimensional aspect of the alignment model made us consider different design choices, all of which had advantages and disadvantages. In the end, however, we did reach a full implementation whose speed was similar to NLTK's algorithm.

The model training was rather uneventful in itself, since although we did initially encounter some bugs, they only became visible in the translation stage.

The translation part was definitely the most problematic of all, since it required every other part to have worked perfectly. Our initial efforts seemed pointless, as both small and large subsets of our corpus seemed to lead to equally bad results. For this reason, we also added the option to use NLTK's algorithm instead of our own, and Wikipedia's training corpus instead of the Bible, so that we can more accurately identify where the errors lied in each of our modules.

Considering the considerable size of our corpus, we initially assumed that the main issue was simply related to insufficient training of the model. Unfortunately, not even using NYU's CIMS servers was fast enough to allow for a very large number of iterations over the entire dataset, which meant that our experiments had to be more limited in scope.

Once all the bugs have been tackled, however, we did manage to get promising results, although they definitely show that much more research needs to be done in this field. In particular, the most important problem seems to be our parallel corpus, since it is not fully sentence-aligned, and the translations are rather inexact. However, the results also show that the method itself can lead to very good results, which will hopefully push research forward in this field.

The actual creation and training of the similarity comparison models were successful. As most of the actual model creation was done by replicating previous work on this topic, we didnt run into too many time consuming issues. Once the model was trained, our comparison algorithm did seem to give slightly inflated scores, however. Our final scoring of the various translation methods we employed were:

Ogdens Manual Translation: 86.82
IBM 1 Automatic Translation: 90.82
IBM 2 Automatic Translation: 90.71

The major difference in similarity between Ogdens own translations and our automatic translations can be explained by the sentence by sentence approach we took. Ogden often did more paraphrasing when simplifying the texts, while we broke them down into much smaller pieces and then swapped out the equivalents. This method complemented the compartmental approach of our similarity comparison module better, and thus gave higher scores. Upon manually reading the texts, the scores are definitely inflated and future work would be in making more accurate comparisons in addition to extending the flexibility of this method to account for paraphrasing as well as more direct substitutions.

References

Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

American Psychological Association. *Publications Manual*. American Psychological Association, Washington, DC, 1983.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

Association for Computing Machinery. In *Computing Reviews*, volume 24, pages 503–512. 1983.

Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK, 1997.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Adrian Sanborn and Jacek Skryzalin. Deep learning for semantic similarity. *CS224d: Deep Learning for Natural Language Processing. Stanford, CA, USA: Stanford University*, 2015.

Luchen Tan, Haotian Zhang, Charles LA Clarke, and Mark D Smucker. Lexical comparison between wikipedia and twitter corpora by using word embeddings. *Volume 2: Short Papers*, page 657, 2015.