Lossy Text Compression Using Natural Language Processing Techniques

Lara Arikan, Thomas Bourne
EE 274 Final Project Report

December 16, 2022

Abstract
The purpose of this project was to explore methods of lossy text compression using techniques already used in natural language processing (NLP). In alignment with nearly all other compression research, the motivation for our project is that humans are generating an increasing amount of data that needs to be stored or transmitted. In the context of textual data, the focus of compression research has generally been in lossless compression. Generally, it can be inferred that lossless techniques are preferred for textual data because the effect text has on the reader changes significantly if certain words or even individual symbols are omitted or incorrect.

Current lossless techniques for compressing textual data are very efficient, and can achieve compression at or near entropy. Lossy compression techniques are often used for media, such as images and audio data, so that compression far below entropy can be achieved while still maintaining the information in an acceptably recognizable state for a human, though not a perfect reconstruction of the original. This project explored the idea that lossy compression of text could substantially improve compression while maintaining the semantic structure of the text. We concluded that it is possible to selectively redact words that carry less of the semantic weight of the text than others, losslessly compress its pruned form to a much smaller size than the original could have achieved, and then reconstruct the decompressed text using either human intuition or recurrent neural networks to an interpretation that is semantically or lexically close to the original.

Background
A set of “semantically non-selective” words, such as “a, an, as, the, of,” are called “stopwords" or “function words"; these words can be excised to improve the performance of sentiment analysis models, or to allow better measures of the semantic similarity of texts using word-by-word comparison (Raghavan et al., 2008, 81). A standard corpus of stopwords has been curated and is distributed within the Natural Language Tool Kit (NLTK), a Python package often used for NLP applications (Sarica & Luo, 2021, 1, 3-4). Our hypothesis was that, because stopwords do not carry a substantial amount of context or semantic understanding, it may be possible that they can be removed and the essential meaning of the text would still be present. If true, an encoder-decoder scheme could be used, along with a lossless compression step in between, to significantly improve compression and then fill in the appropriate stopwords after decompression, though with some incurred distortion. Further, since the “meaning” of a text currently has no definition outside of its effect on a reader, any human being should be able to reconstruct a generally semantically similar sentence from one that is missing its stopwords. In one example of text analysis on fairy tales, NLTK stopwords make up between 75-85% of the text with the percentage increasing logarithmically with the number of words in the text (Sigle & Hvitfeldt, 2021, 3-4). Being able to remove 85% of words from a text while still maintaining its context is optimistic, but shows this could be a significant improvement over losslessly compressing the entirety of the original text.

To reintroduce stopwords algorithmically, a machine learning (ML) model would need to be leveraged. Existing NLP algorithms use vectorization, or embedding, of words to determine similarities between words semantically. The usefulness of an appropriate embedding method would be twofold. First, it would allow us to calculate distortion metrics between different textual excerpts. Second, a reversible embedding method would allow us to train and use a ML model such that we could translate the reconstructed embedding back to textual data.

Sent2vec is similar to the more commonly known word2vec, but more effectively preserves emotional and contextual semantics of sentences than just individually embedding the words (Moghadasi & Zhuang, 2020, 2). Our method requires that we use an embedding that is capable of recognizing semantic differences of sentences as a whole, since context is important to recover the entirety of the textual data we will be excising from and then compressing. Sent2vec is not a reversible embedding process, however, and could only be used for distortion metrics. Sent2vec is available as a Python package with dependencies on other, common Python libraries (Ataee, 2022). Another method that leverages embedding, which we resorted to for reconstruction feasibility, is sequence-to-sequence (seq2seq) translation. In this model, recurrent neural networks (RNN) are used to construct an encoder and a decoder component. The encoder creates a hidden vector which represents each item from the input text along with an abstract representation of its context. The decoder fashions an output item from the hidden vector with use of the embedded context. The “attention” of the seq2seq network can be trained to focus on minor but important features of the text, as an added optimization (Hewitt & Kriz, 2018).

Methods

We removed stopwords in the standard NLTK repository from every sentence in our dataset, which consisted of several chapters of the Book of Genesis from the King James Bible. We thus had about 30,000 sentence pairs - one pruned and one full-length. We vector embedded a fraction of these pruned and original sentences - the first 1000 - using sent2vec, and stored the pruned and original vector pairs as an auxiliary dataset.

To test our hypothesis, and best understand the feasibility of textual reconstruction after using stopword removal, we used three techniques. The first question was whether lexically similar text reconstruction was possible. We knew that the vector embeddings of the original and reconstructed texts should have low cosine distance and low mean square error from each other if the reconstruction were to be lexically similar to the original, since the sent2vec embedding centers around the lexical content of the sentence. We believed that it would be useful to train a supervised machine learning model that took the pruned vectors as input, and output reconstructed vectors to be compared to the original text in embedded form. Even though sent2vec is not reversible, and so we could not access plain reconstructed text in this way, we believed that this would be an important heuristic to determine whether a good reconstruction of the pruned text was possible in a world of embeddings, where some spectral representation of its words would define a sentence, rather than its symbolic makeup. The neural network used to implement this method had one hidden layer of the same vector length as the input and output (768 elements), as the simplest basis to evaluate the feasibility of such an idea.

Another method of lexical reconstruction was to ask people we knew to try and reverse the pruning. The shortened sentences were presented to them with the prompt “Knowing some words have been removed from this sentence, could you write down its original form?” This method was employed because if successful, it was proof of concept that enough semantic information remained in the pruned sentence for its implications to be recovered by the human brain, suggesting that some computational method could be conceived to do the same.

Heartened by positive results in this effort of human reconstruction (described in detail below), we went ahead and implemented the aforementioned computational method: a seq2seq encoder-decoder network that accepted pruned text and output the full-length form. All the hyperparameters of this network (the number of training epochs, the learning rate, the hidden layer size etc.) were taken as is from the PyTorch seq2seq translation tutorial, since we did not have the time and computational resources to tune these parameters (Robertson, 2022). Much of the implementation code of this encoder-decoder network was also taken from this tutorial, although adapted to our purposes. If this method yielded good results, it would function more fully than either above as the decompression module of the lossy compressor.

One final experiment, as suggested to us during the presentation, was to prompt ChatGPT to reintroduce stopwords into the pruned sentences. Naturally, only a few sentences from various texts could be used in this experiment. It was a matter of curiosity to us if such a successful bot could as well as human beings did in text reconstruction, since its ability to generate text is so uncannily similar to human abilities in other applications.

To determine how good the reconstructions were, we decided to use two families of distortion measures. The first were largely to do with the symbolic makeup of the sentences; we used edit distance for the human and seq-2-seq reconstructions, and cosine distance/mean square error for the embeddings. The second was semantic: we decided to employ a mechanical turk to determine how semantically similar people believed the original and reconstructed sentences were, on a scale from 1-10. Similar to the ChatGPT experiment, only a few sentences from various texts could be compared. A variety of texts were used on test subjects with varying exposure to the source literature, to provide a good distribution of data.

For the most part, the main focus of our experiments were around standard NLTK stopwords for removal and reconstruction. We did not gradually remove stopwords to see where a line could be drawn after which reconstruction quality would significantly decrease. However, to demonstrate a correlation between increasing distortion and the number of stopwords removed, we conducted an experiment where only articles (a, an, and the) were excised. Our hypothesis was that the context and semantics of sentences could be almost perfectly preserved with only these words removed, but that this scheme would only minimally increase compression performance.

To measure improvements in compression performance, we losslessly compressed the large bodies of text from where our test sentences came from. The lossless compression standards used were gzip, and zstd at the highest compression level (level 22). We could then compare the size on disk of the original compressed text and the redacted compressed text, to see if these sizes were significantly different.

Results

Our initial steps were to determine the method of sentence embedding in a manner such that the resulting embedding data could be used to train and test our learning model. One concern was that embedding of sentences might result in vectors of different dimensions since sentence structure and length can vary greatly when compared to individual words. After literature review and testing the resulting vectors from sent2vec BERT (Bidirectional Encoder Representations from Transformers) method, we discovered that the embedding will always output vectors of dimension 768. With a fixed length output, this meant that we could proceed with training a model and using it to reconstruct excised sentence vectors.

The realization that sent2vec was non-reversible meant that this was instead used as a testbed for an initial ML model, and also a way to determine sent2vec embedding suitability to measure distortion. Our model was able to reconstruct pruned embeddings to mostly within a mean square error of 0.01 from the original embeddings. There were certain outlying higher-error sentences. (Figure 1) The reconstructed

embeddings were within a mean cosine distance of 0.02 from the original embeddings, which was a full order of magnitude better than the cosine distance between the original embeddings and the pruned sentence embeddings the reconstructions came from. Although these embeddings could not be reverted to readable sentences for qualitative comparison, the experiment suggested that neural networks were a good basic tool to try and create low distortion sentence reconstructions.

The second effort was to use a mechanical turk, essentially just asking humans to reintroduce stopwords from an excised set of sentences. These reconstructions were then compared to the original sentences from which they came and calculated for distortion metrics. In total, four humans were subjected, with varying levels of prior literary knowledge. The distortion metrics were very encouraging in many of the cases, nearly matching the sent2vec model in the previous experiment. Semantic understanding seemed to have much more error and it seems that the stopwords carry more contextual information than originally thought. Table 1 gives average distortion metrics based on the mechanical turk data and Table 3 shows how the sent2vec embeddings improved in cosine distance.

Our third reconstruction method, the seq2seq translation network, was so computationally expensive and time-intensive that we could hardly run a sufficient number of experiments. The tutorial off of which we based our implementation suggested 75000 epochs, which for our dataset led to overfitting - the average loss per trained batch, calculated by the standard negative log likelihood function in PyTorch, started increasing again after 35000 or so epochs. The overall training process took 4.5 hours (Figure 2). We used the suggested random evaluation scheme to see how good the reconstructions were compared to the original: the scheme pulls out 10 sentence pairs and reverses all the special encodings and tokenizations implemented by the decoder, leaving only plain text to be compared. At a glance, its performance is astonishingly bad:

> Pruned version: returned shame face land
> Original version: So he returned with shame of face to his own land.
> Reconstruction: the
>
> Pruned version: thus provoked anger inventions plague broke upon
> Original version: Thus they provoked him to anger with their inventions and the plague broke in upon them.
> Reconstruction: thus of

Our semantic distortion metric, which was to ask people how bad a reconstruction was, confirmed this glancing judgment: participants rated the reconstructions of the seq2seq model almost universally 1 or 0 out of 10 in terms of their semantic closeness to the original. Although we did not have time to try, we have a number of ideas how the seq2seq model could be improved: by feeding it a larger variety of sentences culled not only from one literary mode, but across time periods and styles of literature; by tuning the learning rate and number of epochs to prevent overfitting; and by experimenting with different optimization techniques. Our current decoder makes use of attention, which is a selective weighting of elements in the input that is intended to mimic human attention - how it can isolate and emphasize small details that nevertheless have a large impact on the overall meaning of a text. This function can also be better tuned.

Briefly, we also looked at how well ChatGPT would perform if given a description of the problem and a pruned text, but did not have enough time to develop meaningful distortion metrics. For the most part, ChatGPT would construct a grammatically well defined sentence from the given words, but the meaning was generally lost and the words would be out of order. In comparison of performance to humans, we

observed similar results, but perhaps worse because there seemed to be no way to explain to the program that the words needed to stay in the same order.

In terms of the compressibility of the pruned text, our results were very positive. Using gzip and zstd at level 22 (the highest compression level), our data showed that we could achieve respectively up to 39.7% and 36.6% better compression by removing stopwords. This seemed to increase with text size, correlating with the observation that longer text have more stopwords (Sigle & Hvitfeldt, 2021, 3-4). We also saw that if we only removed articles (a, an, the), compression performance only improved by 7.8% for gzip and 7.6% for zstd on the same text. Unfortunately, we did not have the time and resources to compare only article removal to stopword removal in terms of the ease of reconstruction of lightly pruned sentences relative to more heavily pruned ones, except in human reconstruction. We saw that the edit distance between human reconstructions of lightly pruned sentences (only articles removed) and the original texts were significantly lower than human reconstructions of heavily pruned sentences (stopwords removed).

Finally, we present all our results as a table pairing reconstruction methods and distortion metrics:

Reconstruction method	MSE with original	Edit distance to original	Cosine distance to original	Semantic similarity
Vector embeddings into neural network	<0.01	N/A	Mean value: 0.03	N/A
Human reconstruction	<0.01	~29	Mean value: 0.02	62%
seq2seq	N/A	~59	N/A	3%

Table 1. Pairings of reconstruction methods with distortion metrics.

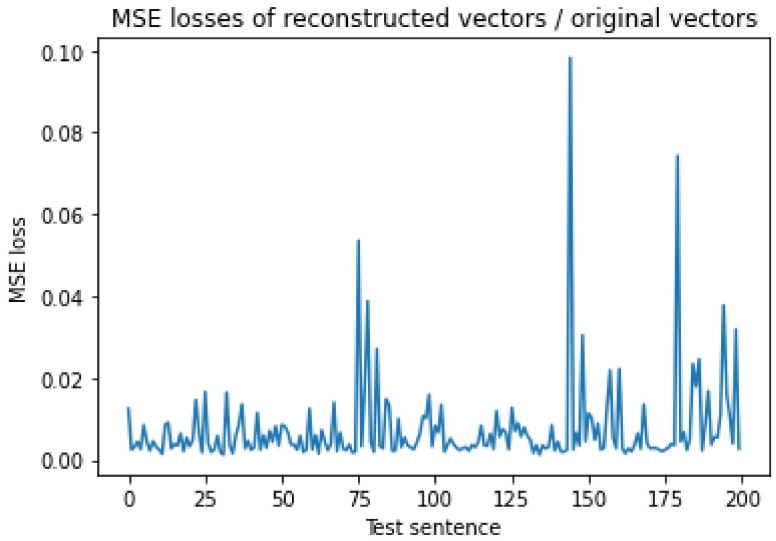

Figure 1: Mean square error between reconstructed vectors output by neural network and embeddings of original text.

```
trainIters(encoder1, attn_decoder1, 75000, print_every=5000)
```

```
17m 35s (- 246m 11s) (5000 6%) 4.7039
35m 43s (- 232m 15s) (10000 13%) 4.3082
53m 41s (- 214m 47s) (15000 20%) 4.0395
71m 58s (- 197m 56s) (20000 26%) 3.9080
90m 30s (- 181m 1s) (25000 33%) 3.7697
109m 18s (- 163m 57s) (30000 40%) 3.7395
127m 56s (- 146m 13s) (35000 46%) 3.6864
146m 31s (- 128m 12s) (40000 53%) 3.6981
165m 3s (- 110m 2s) (45000 60%) 3.7908
182m 53s (- 91m 26s) (50000 66%) 4.3519
200m 9s (- 72m 46s) (55000 73%) 4.6682
217m 26s (- 54m 21s) (60000 80%) 4.4021
234m 43s (- 36m 6s) (65000 86%) 4.2880
252m 20s (- 18m 1s) (70000 93%) 4.3823
270m 27s (- 0m 0s) (75000 100%) 4.2946
```

Figure 2. Train times and average loss during seq2seq training.

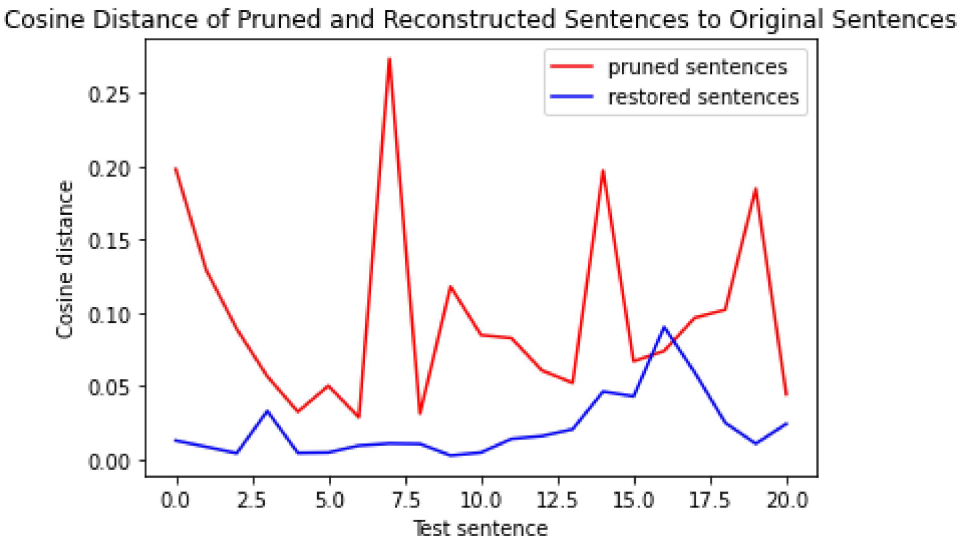

Figure 3. Comparison of cosine distances of human reconstructions and pruned counterparts

Conclusion

More research is needed, but initial experiments point to the possibility that lossy text compression is feasible and beneficial. An embedding technique and ML model specifically built for such a purpose may be necessary, though existing NLP methods could be influential in developing such a tool. The fact that, for certain texts and subject understanding, human reconstructions could be quite accurate at times, indicates that there are contextual metrics that can be conveyed accurately despite significant stopword removal. The compression improvement, particularly for large bodies of text, indicates that this technique is worth pursuing, especially in the context of general textual data that does not need to be perfectly stored or transmitted.

A further experiment could be to determine what are the current limitations of developed ML models for reconstruction of text given an incremental increase in the number of stopwords removed. The list could start with a stopword list of only one word, the compression performance and reconstruction distortion compared, then the list could be increased to two words, etc. An analysis of which words that would be most prudent to excise could be advantageous as well. Ideally, metrics could be determined between which words are most common in a text compared to how much (or little) contextual and semantic information they add to the text. If these two metrics could be quantified and optimized, it would be an excellent way to determine the order of adding to the stopword removal list.

Links

https://github.com/tcmb1987/EE274-lossy-text-compression

References

Ataee, P. (2022, March 23). *sent2vec · PyPI*. PyPI. Retrieved November 25, 2022, from https://pypi.org/project/sent2vec/0.3.0/#files

Gagnon, M., & Da Sylva, L. (2006). Text Compression by Syntactic Pruning. *Advances in Artificial Intelligence*, *4013*, 312-323. https://link-springer-com.stanford.idm.oclc.org/chapter/10.1007/11766247_27#citeas

Gaikwad, D. K., & Mahender, C. N. (2016, March). A Review Paper on Text Summarization. *International Journal of Advanced Research in Computer and Communication Engineering*, *5*(3), 154-160. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ijarcce.com/upload/2016/march-16/IJARCCE%2040.pdf

Hewitt, J., & Kriz, R. (2018). *Sequence-to-Sequence Models* [Lecture]. Stanford University.

Kaur, J., & Buttar, P. K. (2018). A Systematic Review on Stopword Removal Algorithms. *International Journal on Future Revolution in Computer Science & Communication Engineering*, *4*(4), 207-210. https://www.researchgate.net/profile/Preetpal-Buttar/publication/354950829_A_Systematic_Review_on_Stopword_Removal_Algorithms/links/615598e94d9f0f16175f84f1/A-Systematic-Review-on-Stopword-Removal-Algorithms.pdf

Kovář, V., Horák, A., & Kadlec, V. (2008). New Methods for Pruning and Ordering of Syntax Parsing Trees. *Text, Speech and Dialogue - Lecture Notes in Computer Science*, *5246*. https://link.springer.com/chapter/10.1007/978-3-540-87391-4_18#citeas

Merriam-Webster. (n.d.). *Essential vs. Nonessential Clauses: Usage Explained*. Merriam-Webster. Retrieved November 4, 2022, from https://www.merriam-webster.com/words-at-play/usage-of-essential-and-nonessential-clauses

Moghadasi, M. N., & Zhuang, Y. (2020). *Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic*. 10.1109/BigData50022.2020.9378337

Raghavan, P., Manning, C. D., & Schütze, H. (2008). *Introduction to information retrieval* (P. Raghavan & H. Schütze, Eds.). Cambridge University Press.

Robertson, S. (2022). *NLP From Scratch: Translation with a Sequence to Sequence Network and Attention — PyTorch Tutorials 1.13.0+cu117 documentation*. PyTorch. Retrieved December 16, 2022, from https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Sarica, S., & Luo, J. (2021, August 5). Stopwords in technical language processing. *PLoS One*, *16*(8), 1-13. https://doi.org/10.1371/journal. pone.0254937

Sigle, J., & Hvitfeldt, E. (2021). *Supervised Machine Learning for Text Analysis in R*. CRC Press. https://smltar.com/

Zhang, W., Li, P., & Zhu, Q. (2010). Sentiment Classification Based on Syntax Tree Pruning and Tree Kernel. *2010 Seventh Web Information Systems and Applications Conference*, 101-105. https://ieeexplore.ieee.org/abstract/document/5581390