CS230

Cutting Down "Fluff"
A Twist on Text Summarization

Gustavo Torres da Silva (gdasilva)
Kate Salmon (ksalmon1)

Abstract

In this paper we explore the task of making text concise, which we define as keeping all ideas of a text intact in as few words as possible. We built a dataset of 2,225 pairs mapping cluttered text to their concise version, which we used to train our constructed extractive model and already established abstractive summarization model. Our models performed well on the test set, obtaining ROUGE F-Scores of more than 90. The results indicate that using deep-learning to assist humans in writing concisely is worth further exploration.

1 Introduction

Many people struggle with cutting down text for an essay for a class, for a job application and for many other purposes. Writing concisely is difficult, and putting in the effort to cut down text takes time. To solve this problem, we create a model that, given a textual input, outputs a more concise version of that text. Our idea is to do a twist on the text summarization problem. Instead of a model that outputs the "main idea" of the input passage (the typical use-case of text-summarization), we create a model that outputs a concise version of it, keeping all its original ideas intact. We chose this problem due to its novelty, as it is not widely studied, and the potential impact it could have in helping people to write.

2 Related work

The vast majority of work done in summarization consists of capturing the main information in a piece of text rather than conciseness, which we define as keeping all ideas intact in as few words as possible. Our literature review explores existing summarization models that could perform well on our newly defined problem-space.

Extractive Summarization consists of summaries made of the most important words or sentences in a piece of text, which can be done through multiple approaches: reinforcement learning (2), pretrained Bidirectional Encoder Representations from Transformers (BERT) (4) and recurrent neural networks that use multiple different encoders and decoders (3; 5) are some of them. Any of these approaches could be effective for our problem, with the caveat that the extraction has to be at a word level, since a large portion of redundancy lies *within* a sentence (in irrelevant words), rather than *across* sentences.

Abstractive summarization consists of summaries made of newly generated sentences. Like with extractive summarization, there are multiple ways to approach this one. In particular, Abigail See et al. (1) obtained a robust model by combining a sequence-to-sequence attention model that *generates* words with a pointer network that *copies* words from the source text. The approach of generating succinct sentences while making use of the original input to fill in information gaps is also directly applicable to our task.

3 Dataset and Features

We struggled to find a readily available dataset mapping cluttered text to concise text. After brainstorming strategies to obtain such mappings, we came up with a reversed approach, in which we started with the output (concise text) and built our input (cluttered text) from that.

The dataset[1] we used as a basis consists of 2,225 articles from BBC news, with topics ranging from business and politics to sports and technology. We realized that news articles provide a good basis for what we want the output of our model to look like, since they tend to not contain as much excess language as other pieces of writing.

In order to obtain the cluttered version of the BBC news articles, we modified them by adding “fluff” or excess language. We used a Text Inflator[2] to obtain a longer, less concise version of the original text. We wrote a script that copied the complete versions of the BBC articles and sent them to the Text Inflator website, which performed the text inflation and returned “fluffed” versions of the articles back to us to store for use as our inputs.

While this mechanism may not encapsulate all types of excess language, it does a decent job at injecting popular, meaningless phrases that people use (examples include “really”, “very”, “generally”, etc.) at places that make sense and provides a good starting point given that our topic has not been widely studied. More importantly, this strategy allowed us to quickly gather an initial dataset, start training our model and evaluate preliminary results.

The length of our inputs ranged from 122 to 6,208 words, averaging to 523. As for our outputs, they ranged from 89 to 4,432 words, averaging to 384 words. Examples of the "fluffed" inputs and their concise version can be found in Tables 2 and 3.

Once we had our data, we split it up in the following breakdown:

- Train: 1,779 examples (80% of dataset)
- Development: 223 examples (10% of dataset)
- Test: 223 examples (10% of dataset)

4 Methods

We experimented with both extractive and abstractive approaches to summarization.

4.1 Extractive Approach

We initially tried experimenting with an implementation of an extractive model proposed by Y. Zhang et al. (7) that uses convolutional neural networks to learn sentence features and perform sentence ranking jointly by modifying the original CNN model to use a regression process for sentence ranking. Our idea was to modify the model by changing it to perform word ranking instead of sentence ranking since we are concerned with the quality of words within the sentence to generate the concise text. The code base of the model, however, proved to be difficult to incorporate with our dataset and to change it from sentence ranking to word ranking, so we decided to abandon that model and tried to build our own.

The extractive model we built consists of one basic LSTM layer that takes in embedded fluffed text documents and outputs the predicted labels that are then used to generate the predicted concise text. The predicted labels are then converted to binary labels based on a threshold of $p \leq 0.5$. Every word with predicted label 1 was kept and every word with predicted label 0 was removed. We used pretrained word vectors (word2vec) to enhance the performance of our model and binary cross entropy as our loss function. We believed that this was a good approach to tackle our problem since this model would better learn how to distinguish between "concise words" and "fluff words" and would not add any additional or novel words to the document. Lastly, LSTM models are more powerful than other RNN models in dealing with longer chunks of input data, which was necessary for our case because our input documents ranged from 122 to 6,208 words.

[1]https://www.kaggle.com/pariza/bbc-news-summary

[2]http://www.textinflator.com/

4.2 Abstractive Approach

We used the Pointer-Generator (1) as a basis for our abstractive approach. This model combines two mechanisms:

- a sequence-to-sequence attention mechanism that *generates* words for the summary using the vocabulary available.
- a *pointing* mechanism that copies words from the original input. This mechanism allows the model to copy words that are important for the summary but are not available in the vocabulary (e.g. names, numbers and acronyms).

In addition to generating and pointing, the model also includes a Coverage mechanism. This mechanism tackles repetition, which is a common problem in text summarization, in particular in longer texts. For more details on how this model works, see *Get to the point: Summarization with pointer-generator networks* (1).

As a first milestone for our project and as our baseline, we measured the results of a pre-trained model of the Pointer-Generator on our test set.

We then trained the Pointer-Generator (both with and without coverage) on our newly generated train set, setting parameters that better suited to our problem-space (in particular, the maximum length of inputs and outputs).

5 Experiments/Results/Discussion

5.1 Experiments

Extractive Approach. The number of LSTM units was a key parameter to define for finding our optimal extractive model. Typically, increasing the number of LSTM units per hidden layer improves performance, but it can also have diminishing returns and make the model take longer to train. For instance, training one step of a model that has only 1 LSTM unit takes a fraction of the time that it takes to train a model with the standard 128 units. Thus, it was imperative for us to find the optimal number of units that did not significantly sacrifice performance and was time efficient.

Another parameter that was important to adjust was the maximum allowed words for the word embeddings. Since our documents ranged from 122 to 6,208 words, we believed that setting maximum allowed words to be 1,000 would encompass this spread effectively. However, when we evaluated our model, most predictions were near zero making our model output essentially the original fluffed text input (this was due to the fact that there were more texts with less than a 1,000 words, so when we created and padded the word embeddings, the end of most of them were composed of zeroes; we believe this caused our model to be biased towards predicting zeroes). Thus, we experimented with the maximum allowed words for the word embeddings and found 500 words to be the optimal value for our model.

Abstractive Approach. The number of maximum encoding and decoding steps were key parameters to define. On the one hand, the closer these values were to the maximum number of tokens in our dataset (as mentioned in the Dataset section, 6,208 for the input and 4,432 for the output), the more robust the model would be. On the other, the larger these parameters are, the longer the model takes to train (e.g. when encoding steps was 400 and the decoding steps were 120, which is the configuration for the pretrained Pointer-Generator model, a training step took about 20s; however, when we increased them to 800 and 700 respectively, it could take up to 600s). Given this trade-off, we decided to train our model with 800 encoding steps and 700 decoding steps, since that encompassed 89% of our dataset.

5.2 Results

We adopted ROUGE scores(6) as our metric to evaluate model results, which can be found in Table 1. The LSTM outperformed other models in most of the metrics, with the Pointer-Generator with Coverage coming next. The models achieved ROUGE scores of over 90%, which are very promising.

	ROUGE-1			ROUGE-2			ROUGE-L		
Model	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
Baseline	81.34	22.01	33.83	61.60	16.66	25.60	78.48	21.34	32.76
PG	60.67	75.49	60.88	55.69	68.11	55.33	59.14	72.53	58.94
PG + Cov	**94.98**	74.92	81.85	**90.21**	71.20	77.76	**94.12**	74.18	81.07
Basic LSTM	93.36	**90.63**	**91.97**	84.35	**81.87**	**83.09**	92.37	**89.67**	**90.99**

Table 1: ROUGE results for our Baseline (pretrained Pointer-Generator with Coverage), PG (Pointer-Generator without Coverage trained on our dataset), PG + Cov (Pointer-Generator with Coverage trained on our dataset) and the Basic LSTM.

.

5.3 Discussion

In general, both models learned how to identify a large portion of the "fluff" expressions. Most of them were not included in the output generated by our model. Because such expressions were consistent across fluffy inputs, we already expected the models to learn that.

5.3.1 Extractive Approach

The LSTM model does not eliminate all of the fluff expressions from the text, in particular if they consist of more than one word. For instance, the phrase "which is quite significant" was condensed to "which is" by our model, which harmed the meaning of some of the generated sentences.

The state of our model now could potentially be used for something like text-highlighting where it indicates "fluff" words that users should consider removing from their text if they want to be more concise instead of relying solely on our model to produce the concise text in a "ready-to-go" format.

Overall, our extractive model is very basic and pertains to our specific problem, but it could be improved by expanding the the "fluffed expressions" in our dataset and refining it to become more robust to outside inputs.

For an example of the input/output of our basic LSTM model, see Table 2.

5.3.2 Abstractive Approach

The coverage system helped fixing a large portion of repetition issues. A qualitative analysis of the results we obtained from the model without coverage showed that many of the problematic ones were due to multiple repetitions of the exact same sentences (or sequence of sentences). Once we added coverage, many of the originally problematic samples were fixed. The problem, however, wasn't entirely fixed. We can still find samples of outputs that had poor results due to repetition. The majority of such outputs happened in short pieces of text, which indicates that the model might assume a minimum length for the output, which it tries to fill by repeating information.

As a final observation, the model occasionally also removed information from the input. As an example, in a case where there was a quote followed by "said report author mary madden", it removed the latter. We suspect that this is a result of the pretrained model that we trained our model upon, which made use of such strategies to build the summaries.

For illustrative examples of the above, see Table 3.

6 Conclusion/Future Work

Applying established summarization algorithms to a dataset that we created sparked optimistic results for the text conciser task. The results we obtained indicate that it may be very well possible for deep-learning to assist people in making text more concise. As an immediate next step, we should refine our dataset, bringing in cases of "fluff" that our current dataset does not cover. By developing more robust models to remove clutter from language, we will be able to take machine-assisted writing to a new level.

Input	Output Basic LSTM	Reference
Ferguson hails Man Utds **really** resolve Manchester Uniteds Alex Ferguson **literally** has **for all intents and purposes** praised his players gutsy performance in the 10 generally win at Aston Villa,**which is fairly significant**. That **generally** was our **almost** the hardest away game of the season and it was a fantastic game of football, endtoend with lots of **pretty good** passing, said the Old Trafford boss.	ferguson hails man utds resolve manchester uniteds alex ferguson has **for all** praised his players gutsy performance in the 10 win at aston villa **which is** that was our **almost** the away game of the season it was fantastic game of football endtoend with lots of passing said the old trafford boss.	ferguson hails man utds resolve manchester uniteds alex ferguson has praised his players gutsy performance in the 10 win at aston villa that was our hardest away game of the season and it was a fantastic game of football endtoend with lots of good passing said the old trafford boss.

Table 2: Examples of results we got from out test set from the basic LSTM model. Many **fluff terms** were removed by the model, yet **some persisted**. Some of the words that persisted were parts of of a fluffed phrase.

Input	Output (Point-Gen + Cov)	Reference
the film, showing out of competition in berlin, **basically** is nominated for three oscars, including **absolute** best actor for cheadle. sophie okonedo, who **kind of** plays cheadles wife tatiana, **basically** is nominated for **hardly** the best supporting actress **in a basically major way**.	the film , showing out of competition in berlin , is nominated for three oscars , including **absolute** best actor for cheadle . sophie okonedo , who plays cheadles wife tatiana , is nominated for **hardly** the best supporting actress .	the film , showing out of competition in berlin , is nominated for three oscars , including best actor for cheadle . sophie okonedo , who plays cheadle 's wife tatiana , is nominated for best supporting actress .
the study by us researchers, pew internet, suggests musicians **definitely** do not **actually** agree with the tactics adopted by the music industry against filesharing **in a subtle way**. while most considered filesharing as illegal, **really** many disagreed with the lawsuits launched against downloaders in a big way. Even successful artists don't **particularly** think the lawsuits will benefit musicians, **generally** said report author mary wadden **, sort of contrary to popular belief**.	the study by us researchers , pew internet , suggests musicians do not agree with the tactics adopted by the music industry against filesharing . while most considered filesharing as illegal , many disagreed with the lawsuits launched against downloaders . even successful artists don't think the lawsuits will benefit musicians .	the study by us researchers , pew internet , suggests musicians do not agree with the tactics adopted by the music industry against file-sharing . while most considered file-sharing as illegal , many disagreed with the lawsuits launched against downloaders . " even successful artists don't think the lawsuits will benefit musicians , " **said report author mary madden**.

Table 3: Examples of results we got from out test set. We can many **fluff terms** were removed by the model, yet **some persisted**. Conversely, the model sometimes also removed **information that it should have kept**.

7 Contributions

Both team members contributed equally to the project. Kate focused on the scripts to build the dataset and on the extractive model, while Gustavo concentrated on the abstractive model and on the report format.

8 Code

The code for our research as well as the dataset we built can be found at https://github.com/gustavotorresds/fluff-remover.

References

[1] See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

[2] Narayan, S., Cohen, S. B., & Lapata, M. (2018). Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

[3] Cheng, J., & Lapata, M. (2016). Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

[4] Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

[5] Isonuma, M., Fujino, T., Mori, J., Matsuo, Y., & Sakata, I. (2017, September). Extractive summarization using multi-task learning with document classification. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 2101-2110).

[6] Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[7] Y. Zhang et. al. *Extractive Document Summarization Based on Convolutional Neural Networks*, IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society p. 918-922, 2016.

[8] Zhong, M., Liu, P., Wang, D., Qiu, X., & Huang, X. (2019). Searching for Effective Neural Extractive Summarization: What Works and What's Next. *arXiv preprint arXiv:1907.03491*.