THE EFFECTS OF DROPOUT ON NEURAL MODELS FOR ABSTRACTIVE HEADLINE GENERATION

Arjun Bhalla
Cornell University
ab2383@cornell.edu

Ryan Slama
Cornell University
rms427@cornell.edu

Yash Sahota
Cornell University
yss6@cornell.edu

Justin Kuang
Cornell University
zk65@cornell.edu

Abhimanyu Kompella
Cornell University
ak832@cornell.edu

November 26, 2019

ABSTRACT

With the information surge brought forth by the internet and emboldened by the wave of new media, the need to quickly and accurately synthesize and process data is becoming increasingly more important. Automated text summarisation plays an essential role in this process - while extractive methods are well explored, efficient abstractive methods are still not as mature. Currently, neural models perform the best, but can have trouble generalising to the wider world (depending on the nature of the texts). As such, this paper examines the effect of dropout on neural methods for abstractive text summarisation in an effort to provide insight into how to further generalise neural text summarisation methods. Through training well-regarded models with varying degrees of dropout, this paper examined the precision and recall of these metrics over time, finding a dropout rate of about 0.1 - 0.15 to be the optimal for this specific model.

***K*eywords** LSTM · Dropout · Abstractive · Summarization · Machine Learning · Neural Network · Rogue · Headlines · News

1 Introduction

In the modern world, with an increasingly insurmountable amount of information, we are rapidly being overwhelmed with a plethora of important developments which cannot all be physically read. As a result, the ability to accurately and concisely summarise long volumes of text is constantly becoming a more prominent need.

In recent years, the ability of neural networks to perform these natural language (NLP) tasks has increased greatly, particularly in the fields of machine translation, named-entity recognition, semantic role labelling, and language modelling [1]. Specifically, this paper will focus on the task of abstractive text summarisation, which employs an approach of natural language understanding and generation in order to summarise large volumes of text in such a way that new sentences are unique and not seen before within the corpus. This is different to extractive text summarisation, which does an analysis on a body of work and returns a subset of a ranked list of the importance weighted phrases from the text, which serves as the summary.

In this paper, we will primarily be focusing on predicting headline given news articles and blurbs from various media sources. However, the use cases of abstractive text summarization go above and beyond just headline predictions. As capabilities of models improve, the the technique can be used for summarising larger documents. This can be especially useful for legal documents and financial reports. A brief summary of such long documents can potentially save hundreds of hours of work. Such a concept can also aid doctors and other reading intensive professions, by allowing for large amounts of content to be summarized in a concise and precise manner.

A lightweight method that has already seen a lot of success [2], the 4 layer LSTM will be the model of choice in this paper. We will be investigating the effects of varying dropout on the accuracy of the network. An excellent candidate for evaluating these methods are news articles and their respective headlines, because they provide a vast corpus of input (article) - output (headline) pairs for training networks on. Thus, in this paper, we will primarily focus on evaluating the performance of a 4 layer LSTM architecture with varying degrees of dropout in the context of headline generation from a news article.

It is worthy to note that the field of abstractive text summarisation is far from complete. There are multiple issues that must be tackled before we consider the problem to be solved. One of these issues is dealing with out of vocabulary words. Words that are not represented in word vectors but which are seen in the given text excerpt. There have been copy mechanism models that have aimed to tackle this issue. The copy mechanism also aims in reducing the vocabulary of the model as large vocabularies increase the computational requirements for training and deploying the model.

There are other issues too, such as dealing with long streams of text. The issue with long streams of text is that due to the vast amount of input, context and the essence of the article can be easily lost and biased by portions of the article that might not necessarily be relevant.

Another issue that researchers have been working upon is the idea of coming up with better word representations. To tackle this, researchers have used a read-again mechanism that reads the input sequence once to bias the second read that then commits the word representation allowing the hidden vectors to capture more meaning [3]. This is based on the idea of how humans interpret and create summaries. We first glance the piece of information after which we delve into it extract more meaning.

2 Background

The overarching idea behind this method is to first generate a conditional probability distribution - given a certain input sequence of words $x_1x_2x_3...x_n$ (x_i represents some word x) within the corpus, calculating the probability that the next word is each word in the corpus - i.e. $Pr(y_1, y_2, y_3, ..., y_m|x_1x_2...x_n)$. Next, we use a simple policy $\pi(x_1x_2...x_n) = argmax(Pr(y_1, y_2, y_3, ..., y_n|x_1x_2...x_n))$ to determine the next word, where some y_i represents a word in the corpus.

We started by attempting to attain a purely naive baseline - using a 1 layer LSTM network where the input was a sequence of integers that were representative of a word in the corpus. The purpose of this was to iron out any early issues we saw with data processing and parameter choice. We chose not to stem because in attempting to create logical / semantically correct text, the form of the word (i.e. functioning as a noun, adjective, verb (and conjugations), adverb, etc.) is important. We then moved on to a more sophisticated model with word lemmatization (see section 3 about the dataset and its preparation) and an embedding layer.

2.1 Model

The model we used in our experiments began with an embedding layer to encode each word into a 128-bit representation, followed by a 4 layer LSTM with 512 hidden units each, a dropout layer between each (of varying degrees, from 0 to 0.15) with a dense output layer with softmax activation at the end. The motivation for the 4 layer LSTM came from tried and tested results without dropout [2]. We originally chose to extend the dropout to 0.25, but quickly realised that the models past 0.15 were giving distinctively poor results when compared with the others.

We used a standard cross-entropy loss function during the course of training.

2.2 Training

Throughout the course of training we used a method called "teacher forcing", which essentially takes as input a certain subsection of a headline and attempts to train the model to predict the next word in the headline.

However, if we were to just use this method to train, but for testing feed in an article, there would be quite a disconnect between training and testing and the results would quite obviously reflect this. To mitigate this, but to still keep the positive benefits of teacher forced training, we used a policy in which an article would be fed in during training with probability $\epsilon = 0.05$.

We used a learning rate of 0.001 and the ADAM optimiser, with batched examples of 64 per batch. Larger sizes (128, 256) were experimented with but we found that the loss converged at a far higher value when this occurred, so we reduced it to 64. We also used a sequence length of 5.

Each model was trained for 150 epochs, during which the loss was noted and a benchmark of weights was saved every 5 epochs, which could then be retroactively loaded back into the model and used to predict and compare prediction accuracy and metrics. Overall, training for all 6 models took about 4 days on a 12-core CPU cluster.

3 Dataset

3.1 General Information

The dataset used was a subset of the 'all-the-news' dataset found for public use on Kaggle. This consists of 3 large CSV's, each of which contains rows on which headlines, full-text articles, ID numbers, publishers, authors, and dates are stored. We thought it best to attempt to restrict our news to a general domain of politics (instead of also involving domains such as fashion, sports, and entertainment) because then the style of writing, contexts of words, and general tone of pieces and headlines would all be relatively similar in nature, whereas if we had encapsulated more domains there would have been a variety of different words and contexts (especially niche, subject-specific phrases) and it would have overall been quite difficult for our simple model to judge what the context of the article was supposed to be. Thus, we drew from just the first csv file, which contained headlines from The New York Times, CNN, Breitbart, and Business Insider.

3.2 Preprocessing and formatting

The first step we took was to remove punctuation and extraneous characters and words from the dataset - for example, the New York Times articles had "New York Times" in them, which would unfavourably skew the distribution towards favouring those words.

After that, the headline-article pairs were turned into sequences of integers of lemmatized words (using the NLTK WordLemmatizer), each of which represented the index of the word within the vocabulary. Unknown words were ommitted as if they were replaced with a token representing unknown words they would have skewed the network into predicting unknowns a disproportionately large portion of the time.

Finally, headlines were split into input-output pairs for teacher forced training: with a sequence length of 5, we had contiguous 5-length inputs with the next consecutive word encoded as a one-hot vectors as the relative output pair.

4 Evaluation

4.1 Metrics

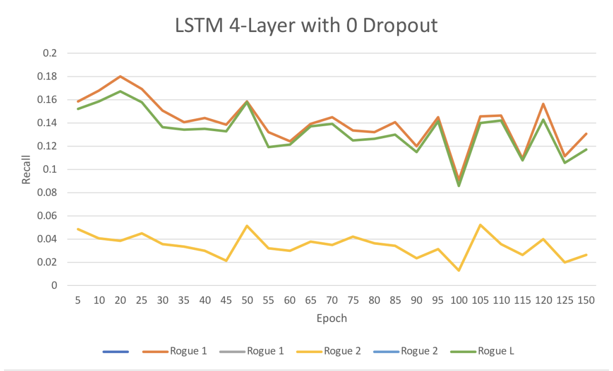
We used rogue 1, 2, and L metrics [4] to measure the semantic similarity of the generated and actual headline. Rogue 1 counts the number of words that appear in both the actual headline and the generated headline. Rogue 2 measures the number of two word sequences, or bi-grams, that appear in both headlines. Rogue L measures the longest common subsequence of words that appear in both headlines. Note that the subsequences do not have to be contiguous.

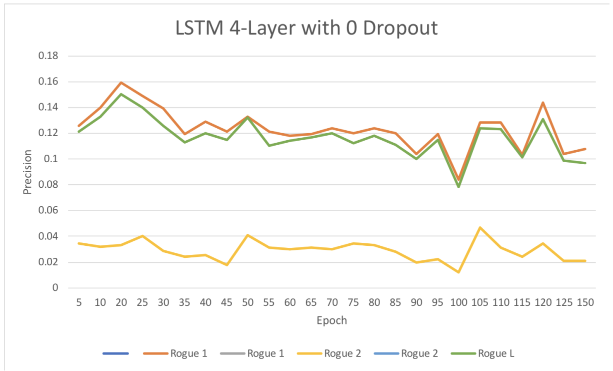
For each metric, we looked at the precision and recall. This is because we want to include all of the correct words from the original title, but at the same time minimize words that are not in the main title.

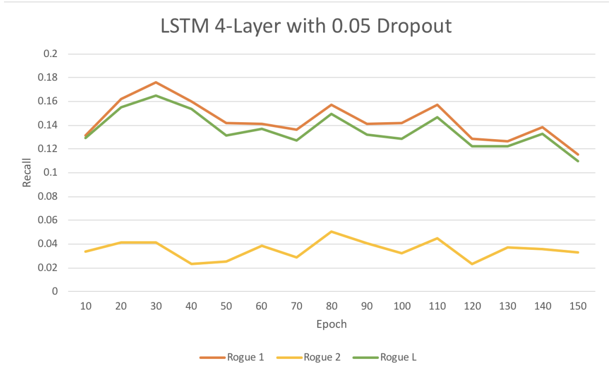
This approach has a glaring issue. Valid headlines do not have to have words in common with the actual headline. For example, The New York Times recently published an article titled “Justices Split Over the Power of Precedent.” Based on the context of an article, the title “Supreme Court Divided Regarding 1979 Decision” would also suffice. That title has zero words in common with the original title and would therefore receive a zero on all of our metrics for both precision and recall.

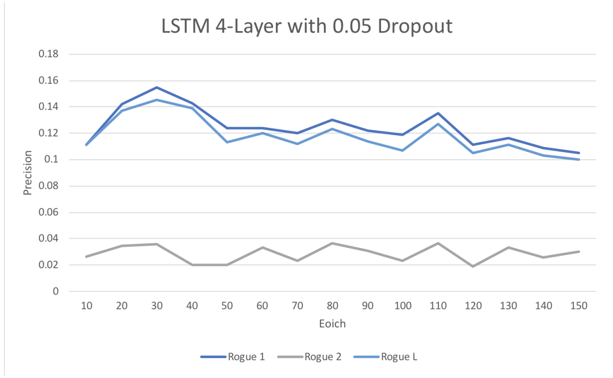
To address this shortcoming, we conducted a survey of humans. We manually unlemmatized several headlines generated by different models and trimmed them to an appropriate length. Then, we created a survey with an even mix of generated headlines and actual headlines after they were processed with out system. Humans believed our headlines to be authentic 44% of the time. We had a false positive rate of 23%, which was due to the fact that our vocabulary size of 20,000 stripped some important words from the headline.

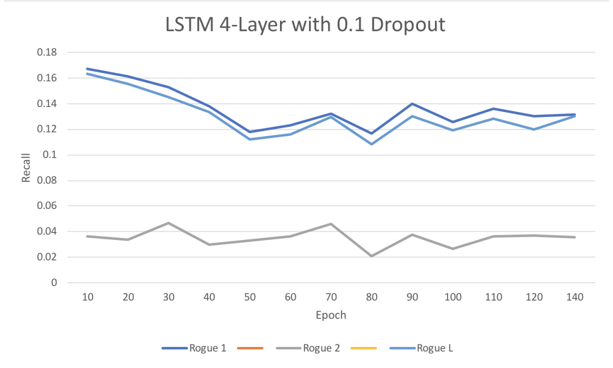
4.2 Results

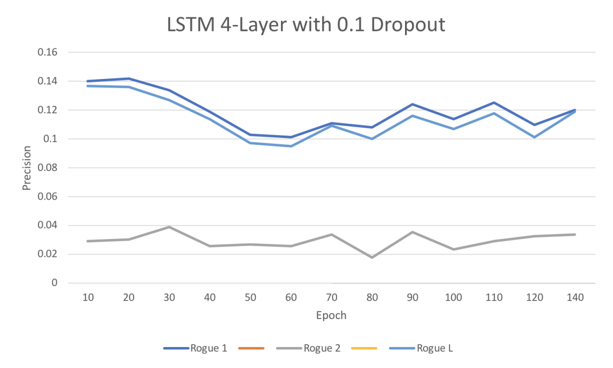

LSTM 4-Layer with 0 Dropout
Recall
Epoch
Rogue 1
Rogue 1
Rogue 2
Rogue 2
Rogue L

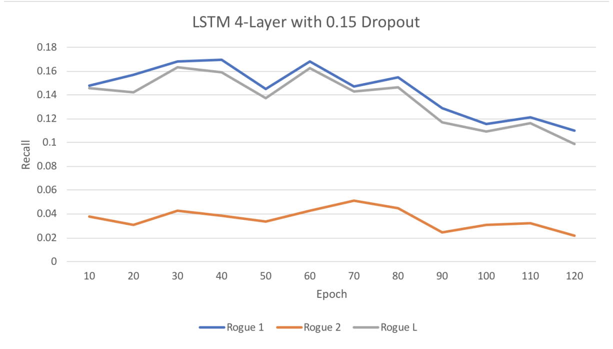

LSTM 4-Layer with 0 Dropout
Precision
Epoch
Rogue 1
Rogue 1
Rogue 2
Rogue 2
Rogue L

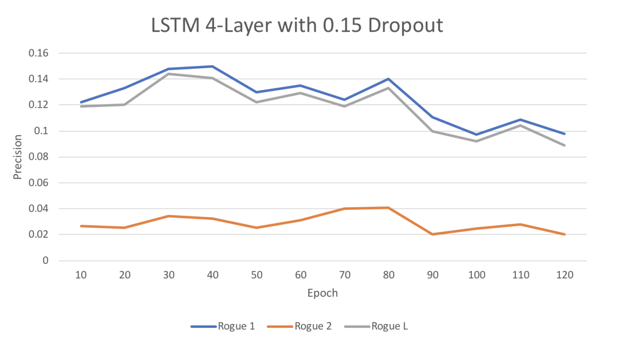

LSTM 4-Layer with 0.05 Dropout
Recall
Epoch
Rogue 1
Rogue 2
Rogue L


LSTM 4-Layer with 0.05 Dropout
Precision
Eoich
Rogue 1
Rogue 2
Rogue L


LSTM 4-Layer with 0.1 Dropout
Recall
Epoch
Rogue 1
Rogue 2
Rogue L


LSTM 4-Layer with 0.1 Dropout
Precision
Epoch
Rogue 1
Rogue 2
Rogue L


LSTM 4-Layer with 0.15 Dropout
Recall
Epoch
Rogue 1
Rogue 2
Rogue L


LSTM 4-Layer with 0.15 Dropout
Precision
Epoch
Rogue 1
Rogue 2
Rogue L

4.3 Examples

Here are some examples of actual vs generated headlines (both lemmatized):

Better Headlines:

Actual headline: florida gov rick scott call on to coalesce behind donald trump
Generated headline: florida gov rick scott threw event focus flight on trump

Actual headline: netanyahu back bill to limit from mosque
Generated headline: israeli prime minister benjamin netanyahu in a to grant to

Actual headline: the fed ha come back to life
Generated headline: federal reserve ha come back to apply of advice afraid

Actual headline: investor are missing the big picture about goldman sachs
Generated headline: most investor look at goldman world in of break in

Worse Headlines:

Actual headline: trump supporter pull rifle on yard sign
Generated headline: a san diego man with the reject reprisal london used

Actual headline: duke and michigan state win head to final four
Generated headline: cnn you'll see some familiar 2016 york you the other

Actual headline: obama explains why it's going to be harder for republican to repeal obamacare than they think
Generated headline: ' ' ' president barack police police for to in

Actual headline: pardoned still held by ice
Generated headline: cnn colorado gov john on year's worker to british the

5 Conclusions and Future Work

5.1 Results

Overall, it appears that the LSTM model with 0.15 dropout at around 40 epochs performed the best in terms of average precision and recall scores on the test set. In terms of what generalised the best, we observe that the highest final ROGUE-n average is found on the LSTM-4 with 0.1 dropout - after 150 epochs, it appears that many models dropped in test accuracy due to overfitting (especially 0 and 0.05 dropout), but 0.1's stayed relatively steady and was the best in both precision and recall out of all the models by the end.

We were able to construct a model that produced adequate headlines for articles that were not too complex. Considering the limited computing power that we had access to for training, it is somewhat surprising how well the current model works at times. There is a lot of room for improvement, but we have shown that there is potential for abstractive text summarisation.

Outside of technical limitations, we also identified issues with the overall domain (political news articles) that can be improved upon.

Something that we understood, but didn't account for was the subjective purpose of headlines. We made the assumption that a headline is supposed to be a summary, and aim to be representative of the content of the article. However, this isn't necessarily true- nowadays many headlines are meant to be "clickbait", and serve to increase interest in the article itself. Although sensationalism in journalism isn't new, the rise of internet-based media has likely increased this trend- even in our dataset (which focused on reputable publications), we manually found headlines such as "Republicans Stonewalled Obama. Now the Ball Is in Their Court" from the New York Times. The article itself primarily focused on the historical context behind the 115th Congress, so our model would not be able to recreate such a rhetorical headline, thus leading to a "bad" metric-based evaluation. It would be interesting to possibly retrain the model on a different dataset that focuses only on sensationalism-based headlines, to see if abstractive text summarization techniques are able to capture non-objectivity.

Similarly, we also didn't account for each article's publisher, author, and date. It would be interesting to see if those secondary pieces of information can be used to improve our model. Different publications often report on the same stories, but with a different focus- being able to account for the uniqueness of each publication would help drive abstractive text summarization forward.

5.2 Future Work

- Addition of attention mechanism
- Testing different models
- Flexible output lengths and variable input lengths

5.3 Attention Mechanism

One of the larger issues that we faced in regards to headline generation was the repetition of words or phrases that were not directly related to the summary of the article. Some headlines contained phrases or names because of the frequency of the phrase or an author's name if they were quoted in multiple portions of the article. Other headlines correctly presented certain details, such as an individual's occupation (Adding President before Trump), but completely excluded most of the main points presented in the article. In order to generate headlines that more closely represented the content of the articles, we are looking to implement an attention mechanism.

Attention mechanisms add what is essentially a "memory-access" mechanism that feeds outputs to the decision making layer based both on repeating sequences (normal RNN's) and by focusing in on a particular aspect of the memory set [5]. This allows for a more context-dependent summarisation by providing words that are more closely related to the article context rather than word occurence. We feel that this would greatly improve the performance of our models and reduce loss without having issues with over-fitting datasets.

5.4 Testing different models

We ended up using an LSTM and experimented with drop-outs to create our current model. In the future, we plan on experimenting with other algorithms, such as using an encoder/decoder network. While LSTM seemed to be the best choice through comparison, it may not perform as well in actuality compared to other algorithms.

5.5 Flexible Output Lengths and Variable Inputs

Currently, our algorithm produces fixed-length (10 word) headline outputs, which is problematic if the article summary does not need to take up the entire length. What frequently results are run-on sentences that have less and less meaning with the continuation of the summary. By allowing flexible output lengths, the article headlines will become both more legible and more accurate. This is a feature that must be added in future iterations of the model.

Currently, our input size is around 150 words, which is less than the length of an average article. Although most of the main article content is usually located at the beginning of the article, we hope to lengthen our maximum input length for a couple reasons. The first is to eventually generalize our algorithm for use on multiple categories of data. The current algorithm is trained to generate headlines for political news, but we would like to be able to generate summaries for more than news (such as legal documents and financial statements).

References

[1] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13:55–75, 2018.

[2] Konstantin Lopyrev. Generating news headlines with recurrent neural networks. 2015.

[3] Sanja Fidler Raquel Urtasun Wenyuan Zeng, Wenjie Luo. Efficient summarization with read-again and copy mechanism. 2017.

[4] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Association for Computational Linguistics*, 2004.

[5] A beginner's guide to attention mechanisms and memory networks. *Skymind*.