Cut to the Chase: An Extractive Approach to Machine Summarization: Final Report

Aykan Ozturk, Andrew Quirk, Ian Hodge

December 16, 2017

1 Introduction

In our digital age, we are constantly bombarded with a seemingly infinite amount of information through sources such as social media, news articles, and instant messaging. Many times, the amount of diverse information (combined with increasingly shorter human attention spans) can lead to an apathy towards reading large amounts of text. This feeling is so common, that it has lead to the creation of the colloquial expression 'tl;dr' (too long didn't read) which refers to short, 3-5 sentence summaries of long sources of information. For our project, our goal is to create an algorithm that will generate readable, and useful, 'tl;dr' summary of a news article, automatically, given the article's original text.

This task can be divided into three main sections:

- Content Selection: Choosing the most relevant sentences from the article.
- Ordering: Deciding which order to put the chosen sentences in.
- Pruning: Remove unnecessary information from the sentences to find the most.

Arguably the most important (and possibly challenging) part of this problem is content selection- deciding which of the sentences will go into the summary. As a result, most of the work that we did was building a useful and tractable model for the analysis of the articles. Once this step had been accomplished, we saw that the order was mostly irrelevant, since the reader doesn't know what order the article was in. Unless the events occurred strictly chronologically, the summaries were strong representations of the article without any tinkering, so we focused our efforts on improving our selection algorithm. Pruning could augment the summaries we generated, and would be a logical next step of the project.

The two main approaches to content selection are unsupervised and supervised learning, and perhaps surprisingly, they yield similar results in general. Recently with more complex deep neural network architectures, supervised learning approaches have started to outperform unsupervised approaches. Also, problems that accompany supervised learning for content selection in general don't really apply to our case as much as other scenarios because our robust dataset provides strong training examples to train the model. As a result, we decided to follow a supervised learning approach to leverage our dataset and get better results.

In the next section we will first describe our model in detail. Then we will describe the necessary preprocessing steps to be able to train our model and learn the parameters. After that we will briefly explain the learning algorithm. Finally, we will present our results and discuss some of the challenges we faced along with a literature review and error analysis.

2 Key Metrics

We will be using ROUGE metrics to evaluate the performance of our models. ROUGE stands for Recall-Oriented Understudy for Gist Evaluation. ROUGE metrics are commonly used to measure the performance of automatic text summarization systems in the literature, so using them allows for comparison with related work. There are multiple evaluation metrics within the ROUGE family, such as ROUGE-N, ROUGE-L and ROUGE-S. The main metric we are planning to focus on is ROUGE-N. ROUGE-N measures the overlap of N-grams between the predicted summary and the reference summary. Assume $N = 2$ for example. In this case, ROUGE-2 generates all bigrams in the generated summary and all bigrams in the reference summary. Then

$$ROUGE2_{recall} = \frac{\text{number of overlapping bigrams}}{\text{total number of bigrams in reference summary}}$$

and

$$ROUGE2_{precision} = \frac{\text{number of overlapping bigrams}}{\text{total number of bigrams in generated summary}}.$$

We can also use the harmonic mean of precision and recall to define a $ROUGE2_{F1}$ score.

3 Data

Our main dataset is a collection of 219,000 Daily Mail articles, along with their human-generated summaries. This dataset is extremely valuable because it provides us with genuine articles and summaries to compare against. Most recent papers on the subject of summarization use this same dataset and the ROUGE framework, so we will be able to sanity check against these results. In particular, the articles include a variety of subject matter and topics, so they provide a good cross-section of human-generated news content to analyze. The summarizations of the articles are also a good evaluative metric because of their differing length and complexity. Some summaries include only simple facts about the article, while others go into a more in-depth analysis of the content. Along those lines, the articles also vary a great deal in length. Longer articles are harder to summarize as a result of the volume of material to cover.

4 GloVe

We used the Stanford GloVe library[5], which provides us with pre-trained word vectors. The GloVe vectors we used were extracted from about 400,000 words from Wikipedia 2014 and Gigaword 5 with dimensionality 100. GloVe scores as high as 75 percent on word analogy tasks and is a good tool to use in analyzing "natural" word ordering. We use this library as a benchmark to construct our summaries in an accurate way.

5 Preprocessing

The first step is preprocessing. There are two main parts of the preprocessing step. We first went through all of the given human-provided summaries, and for each summary looped through each sentence of the article to find which article sentence matched most closely with the summary sentence. In order to determine which sentence most closely resembled the summary sentence, we used the harmonic mean of the ROUGE-2 recall and precision scores. The article sentence with the highest ROUGE score for a given summary was marked as having important content. From here, every article was written to a new file where each sentence was marked with either a '1' for having important content and a '0' otherwise.
Then the second step of preprocessing is to generate a vocabulary file from the dataset by looping over all the documents and detecting unique words. Then, we use this vocabulary file to create a GloVe matrix by finding the GloVe embedding associated with each word in the vocabulary. If there is no GloVe embedding for a given word in our vocabulary, then we assign a random fixed vector to the word (which signifies that it is an unknown word, or an UNK token). Then in training or test time, when we read in an article, we use this GloVe matrix as a lookup table to find the embeddings associated with each word in the article.

6 Model

We consider a dataset of the form (X_{ij}, Y_{ij}) where X_{ij} is sentence j in article i of the dataset, and $Y_{ij} \in \{0, 1\}$ is a binary label which is 1 if X_{ij} is in the summary, 0 if it is not. We want to train a binary classifier (reflex-based model) to be able to infer the label of each sentence X_{ij} from its learned features $\phi(X_{ij})$.
As a sentence is a sequence of words, we first need a way to go from words to numbers (vectors). We used GloVe embeddings[5] to represent words in a sentence as numbers. More information on GloVe embeddings will be given in the following sections. Then we use a neural network on these GloVe embeddings to be able to learn more complex features that don't only treat each word individually, but considers their interactions within each sentence as well. We considered a number of different models for this purpose. The first idea we had was to use an RNN encoder-decoder architecture [1]. This model can be described as follows:

To simplify notation, consider a single article x, and let $x_i \in R^{n \times d}$ be the input vector for sentence i in x after using GloVe embeddings for each word in x_i. Here n is the number of words in x_i (or the maximum possible words in a given sentence to be more specific), and d is the GloVe embedding dimension. In the encoder, we insert x_i into an RNN to encode the sentence information:

$$o_i = RNN(x_i, h_{i0})$$

Here $o_i \in R^{n \times k}$ where k is the state size of our RNN cell. o_i is basically the concatenation of all internal states h_{ij} of the RNN cell, each state corresponding to a single word. Details about the RNN cell will be described in the following sections.
The decoder we used is simpler, we simply have a linear layer followed by a ReLU activation:

$$\phi(x_i)_j = ReLU(o_j W + b_1)$$

where $W \in R^{k \times 1}$ and $b_1 \in R^1$. Here we used parameter sharing: the same b_1 is used for all j instead of defining a vector $b_1 \in R^n$. Our goal in doing this was to build a model that treats a sentence the same way, whether it is the first sentence in the article or the last sentence in the article. Our assumption was that our model should be able to learn the importance of a sentence in article regardless of its position. But defining a vector $b_1 \in R^n$ would also make sense, because for example, first few and last few sentences may actually be slightly more important for the summary so it would have allowed our model to learn this information.
We repeat the same procedure for all words j in the sentence, which means we finally end up with a feature vector $\phi(x_i) \in R^n$.
We feed this feature vector into a linear classifier that computes the probability that sentence i is in the summary as:

$$\hat{y} = \sigma(\phi(x_i)U + b_2)$$

where $U \in R^{n \times 1}$ and $b_2 \in R^1$. The sentence is classified as in the summary if $\hat{y} > 0.5$. We define our content as all sentences from an article that were classified as in the summary. This concludes content selection with the original RNN model. The simple neural model is pretty similar: the only difference is that we use a fully connected (non-recurrent) neural network with ReLU activations instead of an RNN:

$$o_i = x_i W_o + b_o$$

where $W_o \in R^{d \times k}$ and $b_o \in R^k$. The reason why we implemented such a model is because we wanted to be able to compare whether/how using recurrent neural networks improve our performance.
Another change we made on the original model was that we replaced the RNN with an LSTM. LSTMs are more powerful than RNNs in general, they have more parameters and they don't suffer from the vanishing gradient problem as much as RNNs do because of the direct connections they contain. A more formal description of LSTMs will be given in the following sections. Because of these reasons, our intention from the beginning was to play around with LSTMs, and the feedback we got after the progress report also agreed that this would be a good idea. Therefore we trained LSTM-based models instead of RNN-based models. In this case, the model is again very similar except for the encoder:

$$o_i = LSTM(x_i, h_{i0}).$$

One limitation in this initial model is that each sentence is considered on its own. In a text summarization task, the importance of a sentence may depend on the article it appears in. Following this line of thought, we tried including an attention mechanism in our model in an attempt to improve performance as a final improvement on the initial model. Attention mechanisms are widely used in NLP tasks such as question answering. The basic idea behind attention is that we need to refine our representation for the content (article/sentences) based on the task (question) at hand. The problem with text summarization is that there is not an explicit question at hand, the task is defined more abstractly. Therefore one can come up with many different ways to implement such a mechanism, at different levels of the model. The idea we had was to combine the sentence representations $\phi(x_i) \in R^n$ in an article to get a final representation $\phi(\tilde{x}) \in R^k$ for the full article:

$$\phi(\tilde{x}) = LSTM(\phi(X), h_{a,0})$$

where $\phi(X) \in R^{m \times n}$ is the concatenation of all $\phi(x_i) \in R^n$ (so m is the number of sentences in the article). There is one minor difference here compared to the way we used the LSTM while encoding sentences: instead of using the concatenation of all states of the LSTM as $\phi(\tilde{x})$, we only used the final state $h_{a,m}$ of the LSTM, which is the reason why $\phi(\tilde{x}) \in R^k$ instead of $R^{m \times k}$. We are using this $\phi(\tilde{x})$ as we would use a question representation in a question answering task: we use it to reweigh our representations $\phi(x_i)$ for the sentences. Our assumption was that this would be roughly equivalent to asking the following question: "How important is this sentence given the article we are trying to summarize is this article?". This question is exactly the question we are trying to answer in text summarization, which is why we thought it would be a good idea to try this out. In our model, we used bilinear attention, so we compute the score of a sentence x_i

$$s'_i = \phi(x_i) A \phi(\tilde{x})^T$$

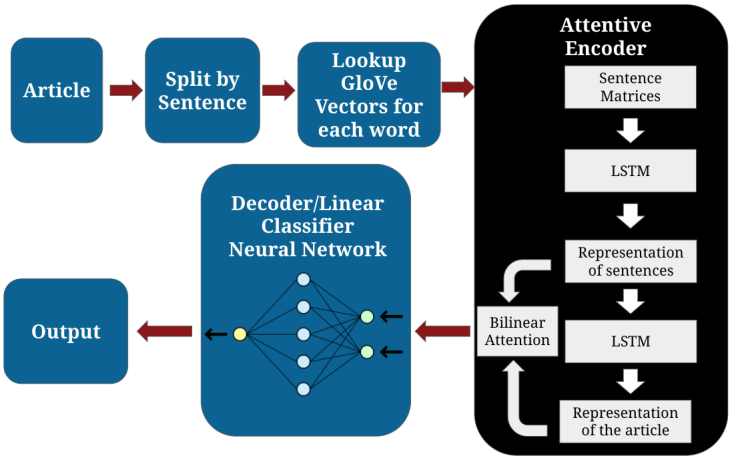
where $A \in R^{n \times k}$ is the attention matrix, and $s'_i \in R^1$. We normalize all scores by using a softmax layer and get a final score s_i for each sentence:

$$s_i = \frac{\exp(s'_i)}{\sum_j \exp(s'_j)}$$

Then we update our sentence feature vectors $\phi(x_i)$ by multiplying them with the scores to get a final representation for each sentence after attention:

$$\phi(x_i) = s_i \phi(x_i).$$

After content selection, the next step is to choose an ordering for the sentences that we selected. We simply used the ordering in the article for simplicity, and saw that this worked well so we didn't change it. Finally there is the pruning step. We found that in some cases, pruning sentences can help create better summaries by removing unessential information and even increasing fluency, but we decided to leave pruning as a possible future work due to the time constraints of the project.



7 Learning

The next step is training, where we read articles sentence by sentence. We have labels for each sentence that we extracted in preprocessing. As we are implementing a binary classifier, we defined our loss function as the minus log likelihood function (or logarithmic loss). We found that due to the fact that our data is unbalanced (there are 7-8 times more sentences with label 0 than there are sentences of label 1), this loss on its own didn't always work well. Therefore we also tried a weighted cross entropy loss where we penalized mistakes on positive labels more heavily than mistakes on negative labels. Similarly, we also tried undersampling negative examples to make the dataset roughly balanced. We applied mini-batch gradient descent instead of SGD. Also we used Adam optimizer[4] in TensorFlow to update weights instead of doing a simple SGD update. The Adam optimizer is different from classical gradient descent in that it leverages both the adaptive gradient algorithm and the root mean square propagation. It uses the exponential moving average of the gradient and the squared gradient, with parameters to control the decay rate of these moving averages.
To prevent overfitting, we regularized our network by using dropout[3] with drop probability 0.2 during training. Dropout optimizes the training algorithm while preventing units from co-adapting more than is necessary. Finally, we used a learning rate of 0.0001.

8 Baseline and Oracle

As a baseline approach, we decided to simplify the text summarization task as a keyword extraction task. In other words, we can generate a simple summary of the text by finding the keywords from an article and outputting them (in some order) without actually trying to form a fluent and meaningful sentence. Of course this is also not an easy task, but it is easier to work on as a baseline. To solve the keyword extraction task, we can simply define a feature vector $\phi(word) \in \mathbb{R}^3$ for each word in the article with the following features

$\phi_0(word) = 1$
$\phi_1(word) =$ Indicator of whether the word appears more than 5 times in the article
$\phi_2(word) =$ Indicator of whether the word appears more than 15 times in the article.

Then we can learn a weight vector $w \in \mathbb{R}^3$ using SGD to build a logistic regression model to classify each word as a keyword or not:

$$p(y(word)) = \sigma(w^T \phi(word))$$

so $\hat{y}(word) = 1$ (keyword) if $p(y(word)) > 0.5$.
After training this model for 50 epochs with a learning rate of .01, we got a ROUGE-2 score of .76 on test data.

We defined two different oracles. For the first, we decided to personally summarize five different articles and calculate the ROUGE-2 score.

Article:

`Geoff Whitington, 63, from Ashford, Kent, was made so sick by obesity-related type 2 diabetes that he had been told he may need a leg amputated`

`(...)`

`Fixing Geoff's diabetes has brought the family closer, says Anthony. 'We went to Scotland over Christmas and Dad was out walking with the grandchildren, which would have been almost impossible two years ago. The kids love having a grandad who is active, and we've got our old dad back.'`

Example DailyMail Summary:

`Geoff Whitington, 63, had diabetes and was on the verge of losing a leg.His sons Anthony and Ian helped the father-of-four shed six stone. He now loves to cycle, never eats take-aways and chooses healthy options.Doctors Mr Whitington is no longer diabetic and he is off medication.`

Human Oracle:

`Geoff Whitington, a dad from Kent, struggled with unhealthy eating patterns. His sons decided to take matters into their own hands and help their father start to eat healthy and exercise. One of the sons decided to make a movie about it called Fixing Dad. Now Geoff is diabetes-free and travels the country speaking about his transformation.`

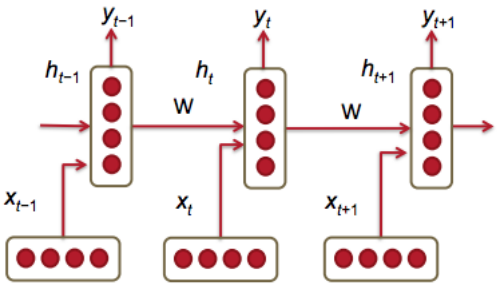
We reported an average ROUGE-2 score of 5.1 for the human-generated summaries.

This gives us a rough estimate for the average human performance (measured by ROUGE metric) on the text summarization task. However, as you can see the human generated summaries often report very low ROUGE-2 scores. As a result, we decided to define an additional Oracle where we evaluated the ROUGE-2 metric assuming that the classifier obtained 100% accuracy. This provided an upper bound on what we could expect from our model, and we obtained a ROUGE-2 score of 26.9.

9 RNN

We used an RNN in our initial model, which uses the state from the previous step and the next word vector in the document to find the next state h_t:

$$h_t = \sigma(h_{t-1}W^{(hh)} + x_t W^{(hx)})$$



Then we can use the state at each time step later to make our predictions. This model provided a good start to what we hope to accomplish and should gave us a good idea of which sentences in the article are the ones we should use for the summary. RNNs sometimes suffer from the vanishing gradient problem, and they are not as powerful as an LSTM or a GRU, which is why we later switched to using LSTMs.

10 LSTM

LSTMs are complex neural network blocks with recurrent connections. Simple RNNs are sometimes insufficient for some tasks due to their limitations. LSTMs are an improvement over RNNs in many different ways. For example, LSTMs are able to "remember" inputs from much far ago (hence the name long short term memory) when compared to RNNs. Also, LSTMs help deal with the vanishing gradient problem seen in RNNs. A very simple explanation for why gradients vanish in an RNN is that things get multiplied with numbers smaller than 1, and as we backpropagate, the gradients decay exponentially as a result. The way LSTMs deal with this is by introducing direct connections to the current state (cell) from the previous state (cell) and the current input. The same idea is also present in GRUs, which are a somewhat simpler version of LSTMs as they have a lower number of parameters and relatively lower expressive power.

There are some different variants of an LSTM, but in our project we used the basic LSTM cell implemented in TensorFlow. This cell's operation can be explained through the following equations:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

where σ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the hidden vector h.

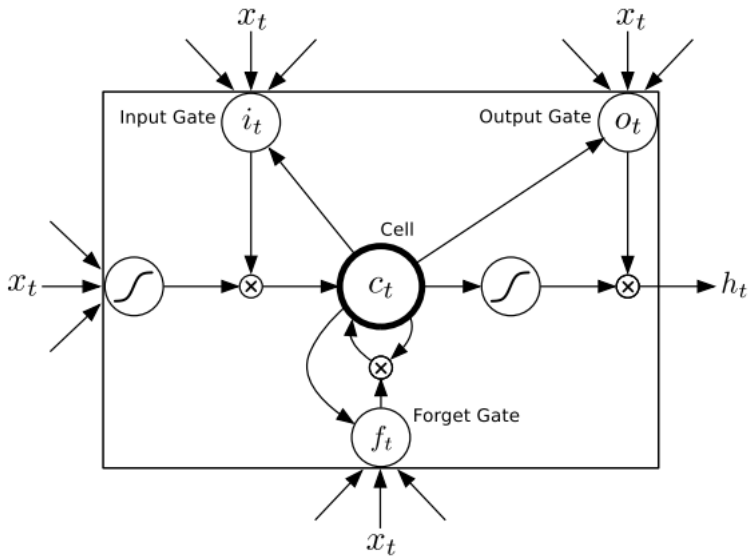


Figure: [2]

11 Results

On the testing dataset, the undersampled LSTM achieved a ROUGE score of 9.7 with an accuracy of 61.8%. The weighted LSTM achieved a ROUGE score of 6.8 and an accuracy of 85%.

Example given Summary:
`Nutritional expert Jackie Lynch defends wartime dishes. Cost-effective and nutritious, army food works just as well today.`

Example weighted-LSTM Produced Summary:
`A new book Bully Beef And Boiled Sweets revisits British military meals since the First World War revealing dishes and ingredients that nowadays rarely grace our dining halls.`

Example given Summary:
`A four metre long carpet snake was found inside a Morayfield home in Queensland. Julia Baker was called to capture the snake that was found on Monday. The snake wrangler believes snakes should not be seen as something to fear.`

Example undersampled-LSTM Summary:
`The large four metre snake was found atop a bed in a home in Morayfield, north of Brisbane in Australia. Julia Baker a British expat has become a snake catcher after falling in love with the reptiles she saw at Australia Zoo 10 years ago`

MODEL	Accuracy %	ROUGE - 2
Baseline	N/A	0.76
Oracle - Human	N/A	5.1
ROUGE-Optimized Oracle	100.0	26.9
NN - simple, undersampled	63.2	9.5
LSTM - weighted	85.0	6.8
LSTM - undersampled	61.8	9.7

12 Challenges

One of the biggest challenges we faced came from our unbalanced dataset. There are approximately 8 times more sentences labeled as unimportant (0) than sentences labeled important (1). Therefore, in our classifier, you can get an accuracy of 85% by producing an empty summary (an output of all 0s) which results in a ROUGE-2 score of 0. Therefore, the high accuracy our classifier might obtain does not necessarily imply a high ROUGE-2 score. We attempted to solve this in a few different ways. Firstly, we tried weighted cross-entropy loss. Basically, we tried to punish wrongly classifying a '1' about three times more than wrongly classifying a '0'. Additionally, we tried under-sampling and made it so that there was an equal number of '0's and '1's in our training set. However, even when we were able to avoid the majority solution, we found that our model still favors brief summaries.
We also believe that there might be a bias problem because none of the models drive training loss sufficiently low. One reason for this might be because there are a lot of unknown words that are not present in GloVe, so we treat all those words as the same UNK token. Also reference summaries are not extractive, so the models that we built is limited in that sense. One th'ng which is interesting to note is that although the LSTM-based model performs best among all models, it's performance is not that much better than the performance of the simpler neural network. Of course, this could be because our assumptions about the model or the data were wrong, or because we had an error in our implementation. But our opinion on this finding is that the main reason for this is because we were unable to train the LSTM-based model for many epochs. LSTM-based model has approximately 10x more parameters than the simple model, but they were trained for the same number of epochs (4). We believe that if we could have trained the LSTM-based model for a longer time (10 epochs maybe), the parameters would eventually converge and we would have much better performance. Of course, the reason why we were unable to do this is because training was taking a lot of time, especially when we worked on the full dataset with LSTM based models. This was another significant challenge as it made it difficult for us to train our model for multiple epochs as well as debug. Our large dataset is helpful in that it can help us achieve accurate findings, but leads to a large challenge when training our model.

13 Error Analysis

We used accuracy and ROUGE-2 as our two main metrics for evaluating performance quantitatively. We achieved best accuracy when we trained our LSTM with weighted cross entropy loss with a weight of 3. But as discussed in the previous part, it is possible to get very high accuracy by a majority classifier that produces

empty summaries. Therefore accuracy is definitely not a reliable metric to use on its own, as we will demonstrate shortly with an example.

The main quantitative metric we used then was the ROUGE-2 metric, as discussed in previous sections. The model with the highest ROUGE-2 score is LSTM with undersampling (with a factor of approximately 8). The summaries produced by this model seem to be pretty good usually, so ROUGE-2 seems to give a good idea about the accuracy of a summary in general. However, we found in analyzing our output that there was not necessarily the highest correlation between ROUGE-2 scores and good summaries. We often saw that summaries that we deemed as acceptable had relatively low ROUGE-2 scores. One major example of this is our human oracle, who only scored a 5.1 ROUGE-2 score despite producing pretty good summaries. Therefore, in this section, we will analyze a few examples where we qualitatively evaluate the performance of our models in different cases. The following summary produced by the weighted model is the first example we will discuss:

Example given Summary:
```
O'Brien spent Presidents Day weekend in
Havana, Cuba, filming the show.  The host
spent days taking in the sights, sounds
and culture of the country.  His show
will air on March 4 at 11 p.m.  Eastern
Standard Time in America.  TBS promises
the trip will give O'Brien's viewers 'a
rare glimpse into Cuba
```

Example LSTM-Weighted Summary:
```
Conan OBrien has become the first late
night host to film an episode of his
show in Cuba since the embargo of 1962.
OBrien spent Presidents Day weekend in
Havana Cuba, filming an upcoming segment
for an episode of his TBS talk show which
will air next month.  Meanwhile, OBrien
reportedly picked the weekend before
Presidents day because his show does not
tape on the holiday, and he would have
been available to travel according to
Deadline.com
```

We all found our generated summary in this case to be quite good, however, it only produced a ROUGE-2 score of 6.4 which is relatively low. This agrees with our previous point.

Because of the unbalanced property previously mentioned, we believed that there was also a tendency for empty or short generated summaries. This can be seen in this one sentence example:

Example given Summary:
```
Amy Lewis, 20, from Manchester, was
told her hands might be amputated.  She
weighed just 4st 7lb at her most ill and
succumbed to other conditions.  One was
autoimmune condition lupus which left her
hands withered.  Now a healthy weight,
Amy has recovered thanks to her passion
for art.
```

Example LSTM-Weighted Summary:
```
Amy Lewis, now 20 from Manchester, said
that her hands were left 'witch-like' and
withered after she contracted a rare form
of auto-immune condition lupus because of
her eating disorder.
```

Empty summaries result in a ROUGE-2 score of 0. Additionally, because most of our data had around 2-4 summary sentences, generated summaries that were just one sentence long tend to result in low ROUGE-2 scores.

As we previously discussed, GloVe vector does not include some proper nouns, and we are forced to represent all unknown words, including these proper nouns, using an unknown token. However, proper nouns and names are often involved in important sentences within articles, and because of that, sometimes our model doesn't recognize the importance of sentences that contains these words. This can be seen in this example:

Example given Summary:
```
The City of Adelaide is the oldest
surviving clipper ship.  It transported
thousands of European settlers to
Australia in 19th century.  Estimates
suggest a quarter of a million South
Australians can trace their origins
back to passengers who travelled on the
ship.  Author Rita Bradd accompanied it
on its final journey around the globe.
A charity dedicated to the ship led a
campaign to save and relocate it.
```

Example LSTM-Weighted Summary:
```
The 70-day voyage took her from Rotterdam
to Norfolk in Virginia, US, and on to
Port Hedland Western Australia, via
the Cape of Good Hope, following the
historic, route the ship', would have
taken in its heyday.
```

It is obvious from the given summary that the subject is the ship The City of Adelaide. However, the name of the ship does not appear at all in our generated summary in part due to the fact that the word 'Adelaide' is

not in the GloVe library.
A final thing that we noticed is that some of our summaries are repetitive. Our model is built only to identify important information, but it has no way of knowing if that same important information has already been chosen. As a result, there were a few generated summaries that simply repeated the same information, like this one:

Example given Summary:

```
Georgina Vilela to serve a minimum of
23 years in prison for murder.  She
must also serve a ten-year sentence
concurrently for arson.  Four of Vilela's
step-daughters were also in the home at
the time.
```

Example LSTM-Weighted Summary:

```
Georgina Pontes Furtado Vilela 47 was
sentenced to 23 years in prison after
being convicted of murder and arson with
intent to endanger life at Wolverhampton
Crown Court earlier today.  Georgina
Pontes Furtado Vilela was found guilty
of murder after setting alight to
her husband Carlos Vilela after she
discovered he had fathered a lovechild
```

14 Literature Review

One distinctive fact about our project is that there has been a great deal of work already published in this field, and many models have out-performed ours in pure ROUGE scores. It is interesting to note, however, that the ROUGE metric is relatively naive and bigram-based, so some summaries that humans would consider "better" might get lower ROUGE scores, and vice versa.

(i) Introduction to the Special Issue on Summarization by Dragomir R. Radev, Eduard Hovy and Kathleen McKeown, published in Computational Linguistics Dec. 2002.

This paper provided a great overview of the different kinds of summarization that are currently being pursued in various contexts. In addition, it provided a great overview of what a "good" summary consists of and led us to use the ROUGE metric. From this paper we were able to see that the model we created is consistent with literature in the field, albeit from years ago. From this paper onward, most research moved from extractive to abstractive summarization, a more difficult and interesting problem, so much of our work was inspired by older papers like this.

(ii) A Deep Reinforced Model for Abstractive Summarization by Romain Paulus, Caiming Xiong and Richard Socher, arXiv preprint arXiv:1705.04304 (2017).

Although this paper focused on abstractive summarization, it provided a great explanation of the RNN encoder-decoder model and showed the upper bound on what to expect for ROUGE scores. Their model scored a phenomenally high 41.16 on the same CNN/DailyMail dataset that we used, and is the next step of what we would try if we were to continue this project. This paper was published this year and is one of the highest ROUGE scores we've seen. Their insights about RNN-based models were taken into account as we developed our project.

(iii) A Survey on Automatic Text Summarization, by Dipanjan Das and Andre F.T. Martins. Literature Survey for the Language and Statistics II course at CMU 4 (2007): 192-195.

This paper also provides great context for our project, and shows the multiple areas that have been pursued in this field. Text summarization has been a major area of research in the NLP community for over 50 years, and this shows that the area we pursued was a reasonable one given the focus of CS221 and our lack of NLP experience. One of the interesting take-aways from this paper was the fact that some of the best-scoring extractive models are simple ones that rely on journalistic conventions by only taking the first sentences of paragraphs and headings where the "important" information is. We wanted to provide a more nuanced summary, and our model did that successfully.

References

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[3] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.