SALTIER: Summarization of Articles using LexRank for Texts via Intermediate Embdedded Representations

Neal Khosla
nealk@cs.stanford.edu

Arushi Raghuvanshi
arushir@cs.stanford.edu

Sasha Sax
sax@cs.stanford.edu

Abstract

Effective text summarization has the potential to allow people to process more information. To further this goal, we present multiple extractive baselines on the NYT dataset, which is a large dataset that immediately yields real-life applications. These baselines combine state-of-the-art techniques from many different sources and provide a solid starting point for further work using this hitherto untapped resource.

1. Introduction

For the first time ever, the average person has to deal with the problem of too much available information. With such a mountain of information online, an important task is therefore to summarize this information into digestible chunks. Effective summarization has been shown to cut decision-making time by half without a significant drop in accuracy [12].

There are three main types of summarization tasks: general, query-based, and update. General summarization is essentially a compression task where at least one document, gets reduced to a shorter textual representation that contains most of the relevant information. On the other hand, query-based summarization is where the user has some external task, and they want a compressed form of the information in the document(s) which is pertinent to a specific query. The summary may be used either on its own or as a tool for estimating document relevance. This task is therefore closely related to information retrieval. The blurbs included in Google results provide an everyday example of query-based summarization. Update summarization is an extension of both of these tasks where the summary aims to provide a concise description of new information to the user. Most efforts have focused on the prior two tasks.

There are several substantial datsets available for the general summarization problem, most commonly used being DUC datasets. With the recent success of deep neural models in natural language tasks, however, it has become important imperative to find even larger datasets. Such systems scale extremely well with the size of the dataset.

Our project will focus on the general summarization task, and we will present the performance of many combinations of extractive techniques on this dataset. In addition, we will present the results from combining these systems in novel ways which appear not to have been considered, or at least not mentioned in the literature.

2. Related Work

In general, approaches create some sentence embedding and then use some other method to select sentences. We found that older work in extractive text summarization focuses on building summaries by finding sentence centrality in some way. Newer work focuses on using neural networks to learn fixed length sentence embeddings, but then uses a very simplistic method to select the sentences for inclusion in the summary. Our goal was to combine on these models by merging state-of-the-art sentence embedding methods with more sophisticated sentence selection methods.

2.1. Sentence Embeddings

A commonly used method for learning fixed length vector representations for sentences is using Auto-Encoders (AEs). AEs are composed of a bidirectional networks with an encoder network layer followed by a decoder network layer. The encoder takes as input a sentence (consisting of multiple word embeddings) and produces a fixed length vector representation. The decoder then takes this fixed length representation and reproduces the input sentence. The goal of the AE is to learn weights such that there is a one to one correspondence between the original inputs and the reconstructed vectors. Then, the embeddings are the fixed length vectors produced by the trained encoder. [1] [3] [4] present different auto-encoder structures.

Another methodology for generating sentence representations is to combine individual word embeddings through a simple summation, averaging, or other metric. Given the current techniques for selecting salient sentences, it is important that sentence embeddings are a fixed length vector.

This precludes the technique of concatenating word vectors and is probably why AEs are so popular.

A third methodology is generating sentence feature vectors is from careful, human-designed feature selection as presented in [6] [10] [9]. These features range from k-means clustering on the text to identifying promising keywords such as 'in conclusion.'

2.2. Sentence Selection

A common technique for sentence selection is to make a sentence-sentence similarity matrix, and then to iteratively pick the sentence that is most representative of all the sentences in to document [5] [3]. This technique is most common in models that used a NN of some sort - probably because it's straightforward and easy, and the papers are focusing on the sentence representation and not on the end result.

TextRank is a page rank based algorithm for sentence extraction [8]. They create a graph where each vertex represents a sentence. Edges between sentences are weighted by the normalized size of the overlap between the two sentences. This allowed them to create a fully connected weighted graph where some edges have very small weights while others have larger weights. Using PageRank on this graph, they selected the top N ranked sentences. One of the largest advantages of this system is that it doesn't need labelled data which makes it highly transferable and reproducible. LexRank is an improvement to the TextRank algorithm. They extract sentences based on their centrality and experiment with different ways of defining the lexical centrality principle.

One problem with these techniques is that they don't account for topic coverage. They pick the most 'representative' sentences even if those topics are already included in the summary. One solution that's been used in the past has been to use Maximum Marginal Relevance (MMR) to select sentences based on a linear combination of importance and novelty, where novelty is how well that sentence is represented in the summary [10].

3. Datasets

3.1. Timeline17

The timeline17 dataset [2] follows 17 newsworthy events such as the BP Deepwater Horizon oil spill over their lifespan. It contains 4650 news articles scraped from web sources such as the BBC. The data is organized by topic and year and contains associated document summaries listed in chronological order to create a 'timeline.' Not all documents for a given topic were summarized.

We parsed the data to retrieve all documents with summaries and gave them a unique identifier. This gave us a total of 1,215 documents with summaries. Since this data is clean and a manageable size, we used it in development.

3.2. New York Times

The New York Times annotated dataset is available through the Linguistic Data Consortium. It contains over 1.8 million articles published by the Times between January 1, 1987 and June 19, 2007. Over 650,000 of these articles include summaries written by library scientists. The article summaries are abstractive and much shorter than the documents themselves. These summaries are sometimes as terse as "HOCKEY", and other times they are simply a copy lead paragraph or article itself.

All of the data is in XML format and is organized by year, month, day, and article id. Due to the incredibly large size of our dataset and limited time to train our models, we used a subset of the data from 2001 - 2005. We believe that our results could be further improved by training on the full dataset. One of the issues with using this dataset is the structure that news articles can take. Some particular issues are articles that are already summaries in some form, articles that are too short to be effectively summarized, or articles that just have brief tidbits on a variety of events, such as sporting articles that just give the result of a variety of matches. Additionally, we faced the issue that generally, the first paragraph of the article contains many of the important details and a high level overview of what the article says - this meant that any system that we had would perform better by simply biasing towards the front of the article. We were able to address this by randomizing the order of the sentences before presenting them to the system.

3.3. Evaluation

Our main evaluation metric for our systems was the ROUGE metric[13]. ROUGE comes in a number of forms, but is a recall oriented metric that attempt to compare the content of the proposed summary to the gold summary. We chose to apply ROUGE-1 and ROUGE-2, n-gram based co-occurence statistics that tell us the percentage of n-gram overlap. The original paper [13] suggests that ROUGE-2 is among the best of all ROUGE flavors, and ROUGE-1 works well, too. Since ROUGE is recall oriented, it inherently is biased towards longer summaries. To counter this, we limit the length of our produced summary to the length of the gold summary. For more specific discussion on our evaluation methods as well as ROUGE as a metric, see our results section.

4. Baseline and Oracle

4.1. Random Baseline as a Lower Bound

As a lower bound for expected performance of our systems, we implemented a method that randomly selects sentences from the original article. This allows us to measure what a system with no intelligence would achieve as a baseline to this problem from which we can estimate the performance of our system.

4.2. Lead Paragraph as an Upper Bound

The summaries in our datasets are human-generated abstractive summaries so it is impossible to recreate them using extractive based methods. This is because news sentences and new words may be introduced in abstractive summaries that do not appear in the document itself. Because ROUGE is affected both by model performance and the nature of the abstractive summary, it can be difficult to compare various systems on an absolute scale.

Fortunately, news articles are written in using the inverted pyramid structure[1] such that the lead paragraph is an overview of the article. Therefore, the lead paragraph serves as the an author-generated extractive summary and is an effective upper bound for how well an extractive summarizer can do. To prevent our models from learning or otherwise abusing this fact, we make sure to shuffle our sentences before feeding them into the summarizer systems.

The lead paragraph oracle allows us to consider the NYT dataset as a extremely large corpus with labeled extractive summaries. This can be used to determine the best techniques for extractive summarization which can then be used on other datasets which do not have a lead paragraph as a built in summary.

5. Approach

A significant component of our project was implementing some of the best methods presented in literature and applying them to a new dataset. In addition, we experimented with combining models in novel ways. Our approach can be broken down into generating sentence representations, applying a similarity metric, and then sentence selection.

5.1. Sentence Representations

5.1.1 TFIDF

Our most classical form of sentence representation was the standard TFIDF metric where each sentence is considered as a document in the term-document matrix. We tried both Porter and Lancaster stemming our inputs, and the difference wasn't statistically significant. Ultimately, we settled on using Porter stemming as Lancaster stemming is quite aggressive. All TFIDF models (unless otherwise noted) use Porter stemming, including the weighting for TFIDF·vector and TFIDF||vector.

[1]https://en.wikipedia.org/wiki/Inverted_pyramid

5.1.2 Vector Sum/Average

Word vectors promise the ability to deal with words in a fuzzy sense. Instead of just looking at word (or synonym) overlap they embed words in a topic space and have nice linear properties. In this approach, we take the word embeddings (GloVE for 100d, 200d and Google News for 300d) of all of the non stopwords in the sentence and take the vector sum or average. Note that the vector sum and average amount to the same results, because the average is simply the normalized version of the sum, and the cosine similarity will give the same relative distances between both.

More formally, for a given sentence s in the document $\mathcal{D}$, the weighted sum becomes.

$$\sum_{\text{word} w \in s} \text{TF}(w, s) \cdot \bar{w}$$

where $\bar{w}$ is the word vector for the word w.

5.1.3 TFIDF-Weighted Vector Average (TFIDF·vector)

Here, we simply calculate a linear combination of the word vectors by weighting each word by its TFIDF weight. Algebraically, we calculate the following:

$$\sum_{\text{word} w \in s} \text{TFIDF}(w, s) \cdot \bar{w}$$

Since TFIDF encapsulates the importance of each word, we hoped that this method would enhance the combination of word vectors by adding some relative weighting to how much each of the respective words contribute to the sentence.

5.1.4 TFIDF Concatenated with Word Vectors (TFIDF||vector)

In this approach, we simply combined the sentence vectors from the TFIDF approach adn the Vector-Sum approach and concatenated them lengthwise to create a new vector. This new vector effectively combines the information from both of the individual approaches and therefore we thought it may provide improvements.

5.1.5 Paragraph Vector

An alternate approach we attempted for creating sentence representations was the Paragraph Vector proposed by Le and Mikolov[7]. This approach is a generalization of the Continuous Bag-of-Words approach for creating word vectors and simply adds in a "paragraph context" token as another word that is generated from a sliding window of words in the current paragraph (or sentence in our case). This approach allows us to learn representations for sentences in

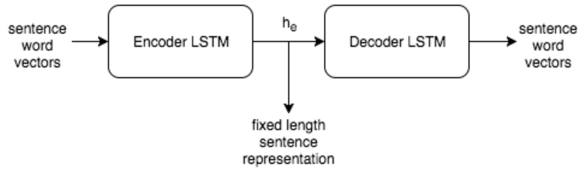

Figure 1. Topology of the autoencoder

an unsupervised manner which can then be applied to the rest of our pipeline. We were able to use the gensim package [11] implementation of this method and run it over our dataset to generate sentence vectors.

The most difficult part of this was parameter tuning as there were a number of different hyperparameters (dimensionality of the generated vectors, learning rate, learning rate decay, minimum number of occurrences per word to be counted in the vocabulary, etc.) that had to be tuned to achieve the best result. Since training took a number of hours per iteration, we employed a random search over the hyperparameters and ran the algorithm overnight before taking the best resulting system. Once we had these parameter values, we trained a model with these values over 5x more data to see if it could achieve better performance with more training data. This increase in training data did present its own challenges and difficulties as we had to switch our system to no longer training in memory.

We note that 200d paragraph-vectors achieve a score similar to 100d GloVe vectors, even though the GloVe vectors were trained on 6 billion words for an unknown amount of time with an unknown amount of compute power while paragraph vectors were trained on a 23.5 million-word corpus on a laptop in 4 hours.

5.1.6 Autoencoder

Many summarizers used autoencoders to compress the sentence to a fixed-dimensional size. We implemented an autoencoder in TensorFlow to experiment with sentence representations learned via deep learning. Both the encoder and decoder were recurrent neural networks (RNNs) with Long Short Term Memory (LSTM) cells. At each time step the encoder takes the next word in the input sentence. The hidden state of the LSTM after the input has been fed in is the sentence embedding. To train the encoder, there is a decoder. The decoder takes as input the final state of the encoder and at each time step outputs a word. The loss function is based on the number of outputed words that were the same as the inputted words which trains the encoder to generate a representation such that there is a one-to-one correspondence between that representation and the words in the input sentence.

Autoencoders are useful because they can learn sentence embeddings in an unsupervised way and without any hand-

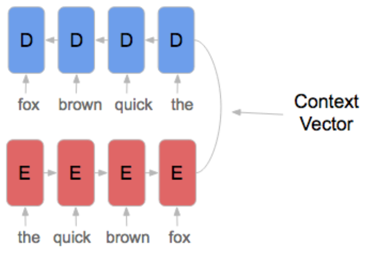

Figure 2. Context Encoder

made feature extraction. This is incredibly powerful given the amount of unlabeled language data that is available on the web.

The downside of autoencoders is that they take a significant amount of time and memory to train, which is a limiting factor given our resource constraints. This is compounded by the fact that that TensorFlow currently unrolls an entire RNN in memory and a nontrivial amount of memory is therefore needed to run an AE on a long sentence. Since news articles tend to have long sentences, mostly over 50 words, this was a factor that we had to take into account.

In addition, there are some critiques that the embeddings from autoencoders more heavily represent the first words in a sentence compared to the later words, since the embedding will produce the first word in the first timestep and will have an increasingly larger number of linear transformations and nonlinearities before producing the later words. To make our training time feasible and to mitigate this critique we propose what we call a context encoder.

5.1.7 Context Encoder

A context encoder is an auto-encoder that takes in a fixed length context windows as inputs instead of sentences. This guarantees that the network will only have to unroll for the context window number of timesteps, and therefore speeds training. We can then sum up all of the context windows vectors that aprrear in a sentence. This also mitigates the criticism that autoencoders only capture the beginning of sentences.

On the other hand, context encoders do not generate an entire sentence embedding using deep learning. In a sense, they are a compromise between word embeddings and unique sentence embeddings.

5.2. Combining Sentences

5.2.1 Cosine

The primary similarity 'metric' we used was the cosine similarity, a standard metric for measuring distance between two vectors defined for two vectors u and v as follows:

$$\cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|}$$

The cosine distance is not a true distance function because it does not satisfy the triangle inequality.

5.3. Euclidean

We also considered a Euclidean distance (induced by the L^2 norm), defined for two vectors u and v as follows:

$$\sqrt{\sum_i (u_i - v_i)^2} = \|u - v\|_2$$

Euclidean distance has the nice properties that it is rotation and translation invariant, and so we thought it could make sense for unit sentences. We made it into a similarity metric by taking $1 - \|u - v\|_2$, and since all the vectors are unit-length this is guaranteed to be nonnegative with similarity equal to one iff $u = v$.

5.4. Other L^p Norms

While other L^p norms could potentially be interesting as distance metrics, we generally viewed these as poor investments of our time and expected little improvement from investigating their applications.

5.5. Sentence Selection

Sentences with the highest maximum marginal relevance (MMR) are iteratively selected for inclusion. It turned out that novelty did not improve scores at all, but we did not know this beforehand.

5.5.1 Maximum Marginal Relevance and the Distance of a Sentence to a Set

Maximum marginal relevance is a way to select sentences for inclusion in the summary based on a linear combination of relevance of the sentences to the document and novelty relative to the summary. Relevance and novelty can be calculated using any function, and distance metrics or similarities are especially well adapted to this purpose. Similarities and distances can usually be interchanged in practice with small adjustments.

The MMR score is formally defined as $\lambda *$ relevance $+ (1 - \lambda) *$ novelty

5.5.2 Hausdorff Distance for Novelty

Hausdorff novelty is a distance metric used to calculate the novelty of a sentence relative to a set summary sentences. There are other novelty functions such as Cross-Sentence Informational Subsumption, but this one based on Hausdorff distance preserves “the no-handcrafted” approach from unsupervised learning. Then let $\mathcal{D}$ be all the sentences in the document and $\mathcal{S}$ all the sentences currently in the summary. Then for a sentence $u \in \mathcal{D}$ $\mathcal{S}$ we define the Hausdorff novelty of u relative to the summary $\mathcal{S}$ as

$$\textsc{HausdorffNovelty}(u) = 1 - \min_{s \in \mathcal{S}} \big(\text{dist}(u, s)\big)$$

5.5.3 Sum Similarity for Relevance

Most papers that we read iteratively picked the most relevant sentence without considering novelty. Relevance scores were calculated calculated as the sum-similarity with all other sentences in the document, and can be considered a measure of centrality of the sentence. Sum similarity is defined as

$$\textsc{TotalSim}(s) = \sum_{s' \in \mathcal{D}} \textsc{sim}(s, s')$$

where $\textsc{sim}$ is a similarity 'metric'. In the papers we read there was no justification of why this relevance estimation was used. This method performed well, often but not always better than LexRank.

5.5.4 PageRank for Relevance

PageRank offers a more sophisticated way to determine relevance scores from a sentence-sentence similarity matrix. LexRank is simple PageRank applied to a sentence-sentence similarty matrix where similarities are the cosine distance between TFIDF-vector representations of sentences. Other sentence representations and other similarity calculations are possible, as described above. Because the original PageRank and LexRank were described only for nonnegative similarities, complications can arise from cosine “distances” yielding negative values.

In general, the PageRank method relies on the transition matrix being stochastic, but negative values make it difficult to normalize each row. PageRank only needs this assumption to guarantee an eigenvector exists, and since our similarity matrix is symmetric we are still guaranteed such an eigenvector by the spectral theorem.

We tested PageRank versus other eigenvector methods, and in practice the unmet stochasticity assumption doesn't create problems. Just in case, we shifted all values by the mimimum in the matrix to guarantee nonnegativity, and this did not strongly affect the scores.

There are many tricks that one can try in order to improve PageRank's effectiveness such as discretizing (setting links to 0 or 1 depending on a threshold), removing diagonal entries, or keeping only the upper triangular matrix in order to bias the results towards earlier sentences. Results got worse for removing diagonal entries, and the upper triangular matrix wasn't useful since we shuffled our sentence order. We did include discretizing in our analysis for comparison to the original LexRank paper.

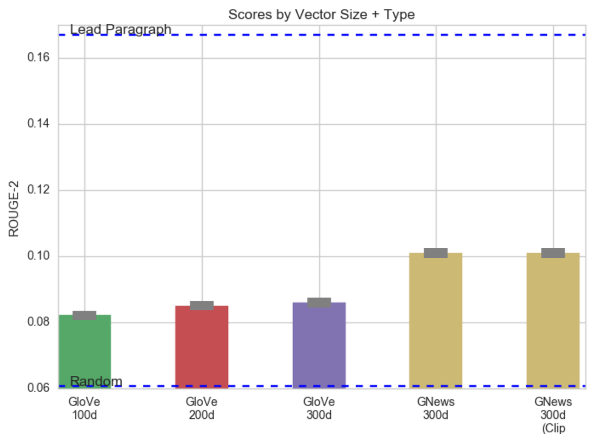

Figure 3. Corpus domain is more important than dimension

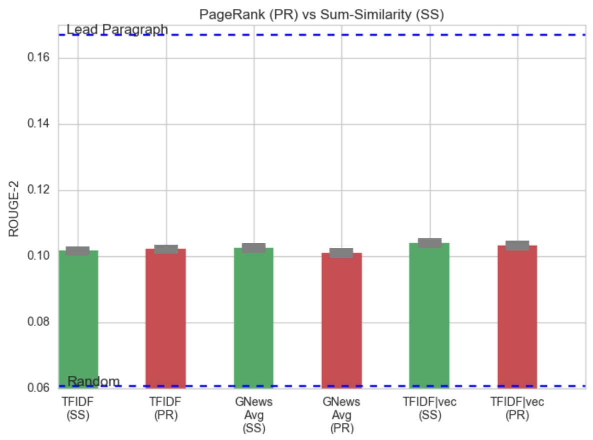

Figure 4. Sum similarity outperforms PageRank for all non-TFIDF models

5.6. Data Preprocessing

To make sure our system is actually summarizing text and not just returning the lead paragraphs, we shuffle the sentences before summarizing. One of the biggest elements of practically implementing our proposed solutions was the data munging and preprocessing we had to do. This primarily included pruning out bad examples such as articles with no content or summaries with no content (even though our system was unsupervised, we used labeled data to measure the accuracy), but also included doing things such as handling non-unicode characters. After pruning our examples, we then parsed all the data files from their natural xml format that included auxiliary information such as the title and date of the article into a readable text format. Once this was done, we could run our pipeline.

6. Evaluation

6.1. Quantitative Results

Table 1 summarizes the performance of our different methods under the ROUGE-1 and ROUGE-2 metrics. Perhaps the most glaringly obvious conclusion from this table is that the different systems do not differ hugely in their performance, as they all have scores between 23.75 and 26.57 for ROUGE-1 and between 8.03 and 10.35 for ROUGE-2. This means that the difference between the best systems and the worst is just about 2% more n-gram overlap. That being said, it is clear that the TFIDF-based methods consistently perform the best across both metrics. In retrospect, this is a reasonable result since the ROUGE evaluation metric favors TFIDF.

More specifically, we can see that the different stemming methods used for TFIDF made very little difference on the results generally. Regardless of the stemming method, these seem to consistently outperform the vector space models. For the vector space models, it is clear that the source and size of the training data made a huge difference in the performance (Fig 3). The models that were trained on the GNews dataset, which was a bigger dataset that more closely resembled our NYT dataset, were better than the standard GloVe vectors regardless of vector dimensionality. However, in general, we notice that increased dimensionality does help the performance of these vector space models.

Another noteworthy feature of these results is that except for the TFIDF model, the models using the sum similarity method for relevance generally performed better than their PageRank counterparts. To us, this was a very unintuitive result which we can only attribute to the possibility that the negative values in the vector models caused issues for the PageRank based methods. Since PageRank relies on a stochastic matrix to find the principle eigenvalue it may work poorly for matrices that are not of this form.

Our auto-encoders were able to approach the optimal perplexity of 1.0 on the Timeline17 dataset within 100 steps of a single epoch. The perplexity plateaued at this value for the following epochs. When running our results on the Timeline17 data, the context auto-encoder approached the scores of the TFIDF model with a ROUGE-1 score of 22.21 compared to the TFIDF score of 23.61. We believe that training on a larger dataset would improve the results of the auto-encoder, particularly on unseen data.

7. Discussion

7.1. Qualitative Analysis of Results

Here, we examine one example of a good output from our system and one example of a bad output and provide some commentary on both. Below is a poor result from our system:

PREDICTED: I once did an interview with a

Table 1. Results

	Performance	
	ROUGE-1	ROUGE-2
Whole Doc	33.04	16.71
Random	21.36	6.07
Max Novelty	24.86	8.61
TFIDF (no stem)	26.62	10.01
TFIDF (Porter)[a]	**26.89**	10.22
TFIDF (Lancaster)	**26.87**	10.23
TFIDF (discrete)	25.53	9.12
Vector Sum Glove 100d	24.02	8.22
Vector Sum Glove 200d	24.35	8.51
Vector Sum Glove 300d	24.40	8.60
Vector Sum Google News 300d (Clip Negative)	26.20	10.11
Vector Sum Google News 300d	26.20	10.11
TFIDF$\oplus$GNews	26.53	10.34
TFIDF$\cdot$GNews	26.06	10.06
Sum Similarity (TFIDF)	26.69	10.16
Sum Similarity (GNews)	26.35	10.24
Sum Similarity (TFIDF$\|\|$GNews)	26.61	**10.40**
TFIDF (Euclidean)	26.39	10.04
GNews Avg (Euclidean)	26.20	10.18
TFIDF$\oplus$GNews (Euclidean)	26.28	10.23
TFIDF$\cdot$GNews (Euclidean)	25.96	10.06
Min Total L^2 (TFIDF)	26.39	10.04
Paragraph Vector	23.94	8.13

[a]All TFIDF will be Porter stemmed unless otherwise indicated

> leading supporter of the movement to privatize – and I asked him, ”Can yo
>
> GOLD: Ray Brady letter on Fred Brock's August 5 article explains why he opposes privatization of Social Security

Here you can see that our predicted summary is almost completely non-sensical. Furthermore, it contains no overlap with the gold summary besides the word "privatize" and "privatization." Unfortunately, this word gives us almost no understanding about what the article is about, especially in the context of the sentence the predicted summary includes. This highlights one of the problems with our system, which is that we cut off sentences at the length of the gold summary. In this case, we can see the full sentence that was selected by our system was actually the following:

> I once did an interview with a leading supporter of the movement to privatize – and I asked him, ”Can you guarantee an annual return to Social Security investors on their money?"

While this wouldn't make a great summary, it certainly suggests a lot more about the article that is closer to the gold summary, that there is some sort of conversation or debate on the privatization of Social Security where the author seems to be opposed to it.

On the other hand, we can see an example where our system performed well below:

> PREDICTED: China exported $125 billion worth of goods to the United States in the first 10 months of last year and imported just $22 billion. But recent excesses – from a frenzy of factory construction to speculative inflows of cash to soaring growth in bank loans – suggest that China may be in a bubble now, especially on the investment side of the economy. But any significant broadening could slow the Chinese economy in a hurry, and with it the economies of many Asian neighbors that increasingly send components to China for final assembly and reshipment to the United States. If the central bank actually sold extra bonds as fast as it bought
>
> GOLD: Recent excesses in China's economy, in-

> cluding frenzy of factory construction, speculative inflows of cash and soaring growth in bank loans, raise concerns that China may be in bubble, especially on investment side of economy; most economists predict that growth will have to slow sometime this year, at least for investment side of economy; say China is likely to rely on consumer spending to prevent any slowdown from becoming too severe; assess risks posed by possible cyclical bust in investment sector, intensified trade protectionsim in US, banking crisis and political unrest or some other loss of social stability; photos; graphs (L)

Here we can see our system not only demonstrates a good overall summary of the concepts in the gold summary, but also shows a fairly high level of fluency. This type of result is encouraging in that it highlights what extractive summarization looks like when it works properly. That being said, examples like this were few and far between in our results. Many of the summaries which achieved the highest ROUGE scores were very poor summaries from a human evaluation standpoint. On the other hand, some summaries with lower scores, were relatively more fluent than those with higher scores. This is largely because ROUGE doesn't consider language fluency at all, highlighting one of a few problems with the ROUGE metric.

7.2. ROUGE as a Metric

While ROUGE is the most common metric for measuring performance of summarization systems, it has a number of limitations that are worth discussing. First off, as mentioned, the metric is recall oriented, which means it will automatically favor longer summaries. We were able to address this issue by limiting the length of our summaries to the length of the gold summary for each data point. Additionally, ROUGE-1 and ROUGE-2 are based on n-gram co-occurence. While this is fine in theory, it does mean that ROUGE measures total word overlap between our summary and the gold summary. This is problematic for extractive systems since the gold summaries are abstractive while our summaries are extractive. This places an artificial bound on our system's accuracy since we cannot ever hope to include words that are not in the original article in our summary.

Furthermore, this means that ROUGE is brittle and fails to measure any amount of semantic understanding as it follows something much closer to a bag of words approach for comparing summaries than anything else. This also presents problems for evaluating our systems as news articles are generally thematic and so our summarization system can get credit for including a words that are frequently found throughout the article. For example, an article on the Iraq war would almost certainly contain the words "Iraq" and "war" in the gold summary and these would probably appear in our generated summary even if the sentences they were included in did not capture the overall message of the article. Since the upper bound on our systems is estimated by our lead paragraph method to be somewhere around 33.04 ROUGE-1, this means we can achieve a good system by just including one out of every three words of the gold summary in our generated summary. By looking at the score of our random system, we can see that this is actually problematic, as the random system can achieve a ROUGE-1 score of 21.36, which approaches the range of our actual systems.

Finally, ROUGE does not measure summary fluency. A "perfect" summary could contain all the words necessary and get a ROUGE score of 100, but the sentences could be reordered into an order that renders the text completely incoherent. Try rearranging the sentences in this paper and see if it still makes sense!

7.3. Resource limitations

We ran into both time and memory resource limitations when training our auto encoders. We found that existing systems with 2 million sentences, a 2 layer LSTM, and a softmax vocabulary of 80,000 take about 2 days to train. It would be academically satisfying to use a full-strength system for this project, but given that our full NYT dataset contained over 200 million sentences, and a vocabulary of over 400,000 words, it was simply not feasible to build, debug, train an auto encoder on our entire dataset, and tune parameters in the time-frame of a class project.

As a result, we took trained on the Timeline17 dataset, and limited the vocabulary to 10,000 words. This allowed us to generate some preliminary results on the auto-encoder, however we believe that they would be much better if we had the time and computational power to train them on a larger portion of the dataset.

8. Future Work

8.1. Evaluation Metric

As discussed above, the ROUGE evaluation metric has many problems. A better evaluation metric would take synonyms and fluency into account.

A more general problem with extractive summarization is that the best sentences to summarize a document my not flow well together if simply concatenated. It might be interesting to either 1) focus more on abstractive summarization, or 2) develop a system to order and add transitions to extractive summaries to make them more readable. It is important to then have an evaluation metric which rewards well worded summaries.

8.2. End-to-End Systems

We noticed that all of the work that used deep learning for summarization were focused on the subtask of generating sentence representations. However, it is unclear if the auto-encoder objective is the best objective for the summarization task.

We believe that a better approach would be to train an end-to-end system that optimizes a full network based on the final summarization score. One such approach would be to use a variation of a dynamic memory network.

Dynamic memory networks (DMNs) have recently shown very promising success in the Question Answering field. We can formulate summarization as a question answering task where the input is the document and the question is 'what is the summary of this document?'. DMNs have an input, question, answer, and episodic memory module, and all modules would be trained based on the final summary. We believe that experimenting with these types of end-to-end systems may lead to better results.

9. Conclusion

Extractive text summarization is largely limited the poor evaluation metrics and a lack of labeled data. Most datasets available have abstractive summaries instead of extractive ones. While we demonstrated some possible improvements over old extractive summarization techniques, it is apparent to us why this task was abandoned by the DUC and why there has been little progress on it recently. Namely, it is a poorly defined task where quality data is difficult to find and evaluation is both difficult and potentially does not reflect the desired outcomes of an ideal system (i.e., inclusion of important semantic concepts from the text in the summary).

References

[1] M. Y. Azar, K. Sirts, D. M. Aliod, and L. Hamey. Query-based single document summarization using an ensemble noisy auto-encoder. *Proceedings of Australasian Language Technology Association Workshop*, pages 2–10, 2015.

[2] N. T. M. A. G. B. Tran, T.A. Tran and N. Kanhabua. Leverage learning to rank in an optimization framework for timeline summarization. *TAIA workshop, SIGIR*.

[3] B. Hu, Q. Chen, and F. Zhu. LCSTS: A large scale chinese short text summarization dataset. *CoRR*, abs/1506.05865, 2015.

[4] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. pages 31–39, 2014.

[5] K. Kaikhah. Text summarization using neural networks. 2004.

[6] K. Kianmehr, S. Gao, J. Attari, M. M. Rahman, K. Akomeah, R. Alhajj, J. Rokne, and K. Barker. Text summarization techniques: Svm versus neural networks. pages 487–491, 2009.

[7] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[8] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts.

[9] R. Mihalcea and J. Wiebe. Simcompass: Using deep learning word embeddings to assess cross-level similarity. *SemEval 2014*, page 560, 2014.

[10] A. Nenkova, S. Maskey, and Y. Liu. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, page 3. Association for Computational Linguistics, 2011.

[11] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[12] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 2–10, New York, NY, USA, 1998. ACM.

[13] C. yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.

10. Appendix

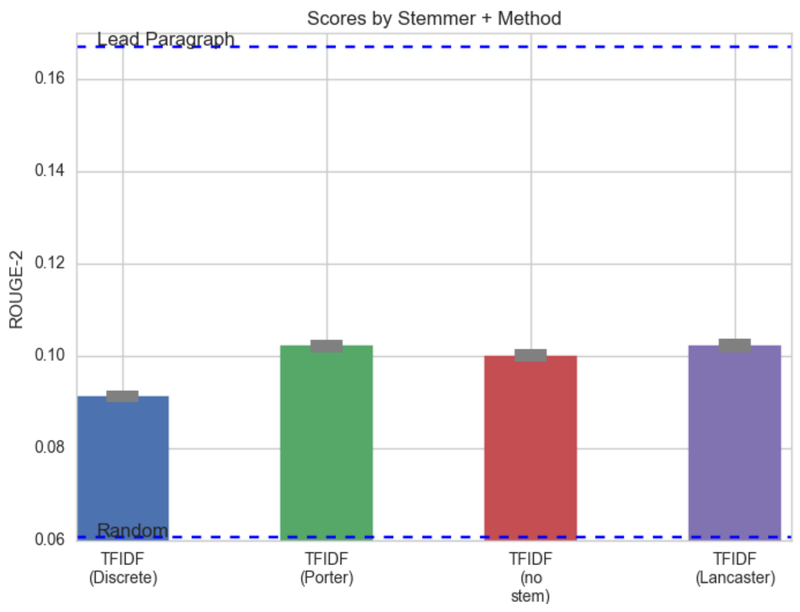

Figure 5. Stemming helps a bit

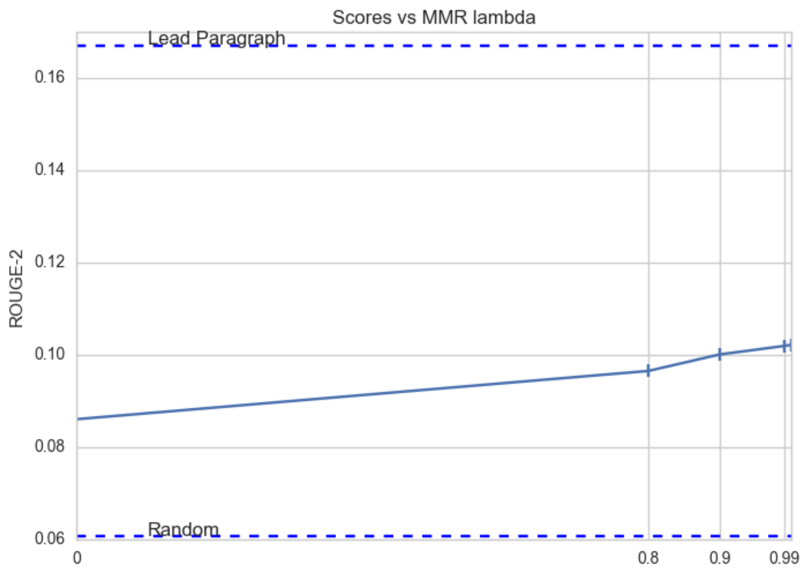

Figure 6. MMR doesn’t help

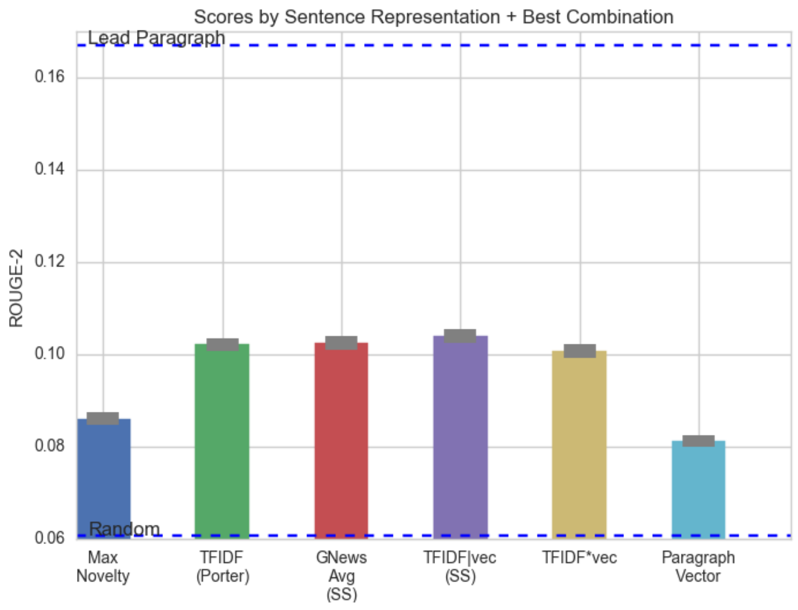

Figure 7. Comparison of the best combinations for each model

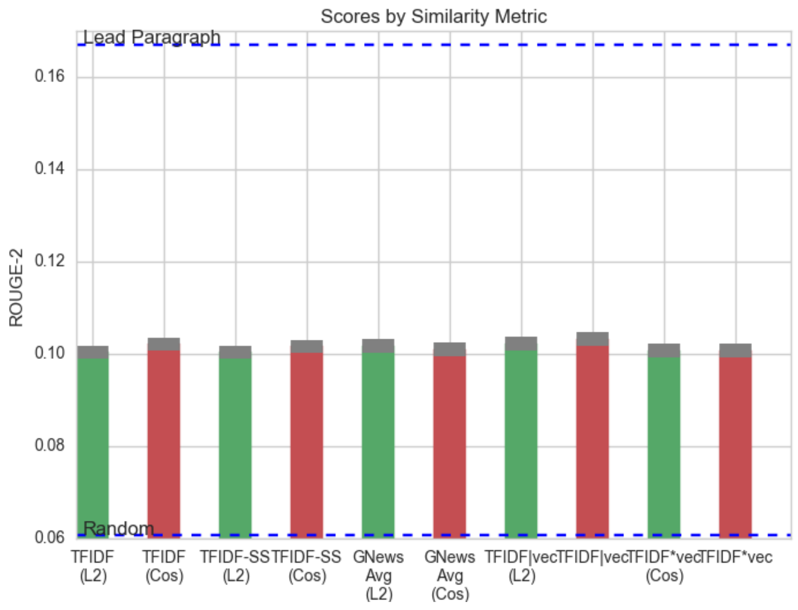

Figure 8. Cosine distance almost always does better than Euclidean