Project Report for TDDE16 Text Mining

Evaluation of Unsupervised Text Summarization Algorithms

Using Pre-Trained Sentence Embeddings and TFIDF

Viktor Gustafsson

vikgu708
Linköpings universitet

HT, 2020

Abstract

This work investigates the potential performance gains of using sentence embeddings from S-BERT when using extractive unsupervised text summarization models such as LSA, Clustering and TextRank compared with using TFIDF. The CNN/Daily Mail dataset is used for evaluation with ROUGE scores as metrics. The performance of these models are also measured over the number of sentences allowed in the summary, to study how well the algorithms scales with summary length.

Models that utilize sentence similarity as the main metric for summary generation, such as Clustering, sees the largest performance increase at around 20% compared to using TFIDF. However, the models that performs better with TFIDF, TextRank and Weighted LSA, still performs best on the dataset. TextRank performs better on shorter summaries, while weighted LSA performs better on summaries with four sentences.

The code for the project can be found in the author's github at viktorgus/tdde16-proj.

Contents

1. Introduction

1.1 Background

Text summarization is the act of shortening a target text substantially whilst still covering the main ideas in the text. Summarizations are necessary in many contexts, for instance when a reader wishes to quickly skim the main contents of a news article. Since manual text summarization can be quite costly, a system for automatic text summarization can result in significant cost savings.

In recent times the NLP community has seen performance increases in many areas, including text summarization, due to the advances in Deep Learning research. More specifically, the usage of transformers in NLP has lead to models such as BERT outperforming previous SOTAs for many tasks. These improved models has also lead to Word-Embeddings from these models outperforming previous word-embeddings on task such as STS. One noticeable variant of this is S-BERT, that encodes entire sentences in vectors and achieves SOTA benchmark on STS tasks.

1.2 Motivation

Most summarization SOTAs for different datasets are achieved with transformer models, for instance MatchSum [10]. SOTA performance is usually achieved with supervised models i.e. models that need labelled data for training. A problem arises when one needs to summarize data from an unlabelled dataset, as Deep Learning models typically needs a considerable amount of data to achieve great performance, and manual summarization is typically an expensive task. In some cases, there might exist a labelled dataset similar to the intended task for the model, in which case a model can simply be trained on this dataset. However, in some cases, the summarization data might be novel and lack a corresponding labelled dataset.

Therefor, the need for unsupervised algorithms still remains. Unsupervised text summarization typically uses matrix factorization or graph ranking models over a measure of similarity of sentences. When these algorithms were proposed, pre-trained word embeddings from models such as BERT were not available. Instead, most papers for these algorithms use similarity measure based on TFIDF or other traditional word embeddings. One natural question therefore is if these algorithms can achieve greater results by leveraging the recent advancements in Deep Learning for NLP. Since word-embeddings from BERT has proven to outperform other embeddings and TFIDF on sentence similarity tasks, the hypothesis is that this should be the case. The masked token and next sentence prediction tasks used for pre-training the BERT models, and in conjunction the word-embeddings, are very general, using text datasets from a very wide range of sources. This means that BERT-based embeddings should be general enough to use even in novel summarization tasks, but might be limited by specific domain knowledge.

Can unsupervised extractive text summarization algorithms achieve higher performance with the use of pre-trained word embeddings from transformer models, specifically S-BERT?

1.3 Research Questions

This work aims to investigate the possible performance gains on a set of text summarization models by the usage of S-BERT word embeddings for a more precise measure of semantic sentence similarity. We also study how the models perform with different number of sentences in the summary. The evaluation uses F1 measure of Rouge-1, Rouge-2 and Rouge-L between the model's summary and the gold summary from the dataset. The evaluated models are:

- TextRank
- MatchSum-U.
- LSA
- Clustering

These models are compared to a set of baselines that will provide an indication of relative performance. These are:

- Oracle: An optimistic bound for the ROUGE metrics
- Random: A pessimistic bound for the ROUGE metrics
- Lead: A "well performing" baseline

The models are defined further under section 4.2.

2. Theory

2.1 BERT

BERT, short for Bidirectional Encoder Representations from Transformers is a transformer based model proposed by Devlin et al. [2]. BERT and models built upon the BERT model has pre-trained checkpoints available via the hugging face distributed transformers and sentence_transformers libraries, which is utilized in this work. BERT is versatile and can be used for many NLP tasks, Information Extraction, Text Classification, Summarization, Text Generation and more, when fine-tuned on specific data. This is due to BERTs pre-training tasks of Masked Language Modelling I.E predicting masked words in sentences, and Next Sentence Prediction. Before BERT, most state of the art language models used some form of RNN, such as LSTM or GRU networks. The main issue with these models that BERT solves is that the final output of the encoder blocks' hidden states encodes all previous inputs, and thus long term relationships are difficult to maintain and encode. It is also difficult to use gradient descent on such models, as the models are very deep and thus the gradient descent is usually truncated to a few model units. BERT avoids both of these problems and enables faster parallel computation and training due to the fact that all input words are processed in parallel thanks to the attention blocks. The attention blocks, in short, are trained to link relationships between words with the self attention block, and then links relationship to the output sequence with the multi-head attention block. This results in more efficient and faster training.

Word embeddings can be extracted from the encoding layers of BERT, and these embeddings gain semantic meaning depending on context due to the attention mechanism. For instance the word "bank" can either refer to the land alongside a river or a financial institution. Previous word-embedding models such as GloVe did not encode such context dependent semantic meaning.

2.2 S-Bert

Previous models for sentence embeddings based on BERT usually either used the average word embeddings from sentences, or the next sentence output for the CLS token. In practice these sentence embeddings usually perform worse than previous embedding models such as GloVe. S-BERT, a model proposed by Reimers et al. [8] aims to solve this issue. S-BERT uses siamese BERTS (two BERT models with identical weights). S-BERT is pre-trained on the Stanford distributed NLI data which measures semantic textual similarity from human judgements. S-Bert outperforms many pervious SOTA models such as average BERT and GloVe on the STS tasks from STSbenchmark.

2.3 TextRank

TextRank is an unsupervised algorithm that ranks text by importance in a weighted graph with an iterative voting scheme algorithm. TextRank is based on the algorithm PageRank, famously used in the Google Search Engine. The basic idea of PageRank is to conceptualize websites as nodes and links as edges in a directed graph. Then websites are ranked by importance by an iterative voting scheme, where each website that links to the target website 'votes' on the target websites importance. This idea can be extended to an unidrected weighted graph, where the votes are now based on node importance as well as edge weights. This concept is used in TextRank as proposed by Mihalcea et al. [5]. For our purposes, nodes are sentences and edge weights are the sentence similarity score for the sentence pairs.

$WS(V_i)$: Rank for Sentence i under current iteration.

W_{ji} : Similarity for sentence ij, computed with cosine similarity of the sentence embeddings (The original paper uses a word similarity measure instead, so using sentence embeddings that scores high in Semantic Text Similarity tasks such as S-BERT should hypothetically produce performance increases).

$ln(V)$: Set of all neighbouring Sentences (non-zero similarity) to V.

d : Damping factor. Often conceptualized as the "probability that a user clicks a link" in the original PageRank algorithm.

$$WS(V_i) = (1 - d) + d * \sum_{Vj \in ln(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \tag{2.1}$$

The recursive computation above is repeated a fix amount of iterations to ensure convergence.

2.4 LSA

LSA is an unsupervised algorithm useful for detecting hidden semantic structures in text data. LSA traditionally uses word vectors such as TFIDF to extract information such as which words are often used together, and which sentences share similarities in word usage. LSA uses Singular Value decomposition to factorize the matrix of word vectors into a word-topic matrix M, a topic-sentence matrix V, and a diagonal matrix Σ that contains the weights for the topics:

$$S = U \Sigma V^t \tag{2.2}$$

Intuitively, Σ describes how 'important' each topic found is in the S Matrix, and is the main focus for LSA analysis. [7]

3. Data

The text summarization algorithms are all evaluated on the CNN/Daily Mail dataset. This dataset is common for extractive summarization, and is therefore useful for comparison for similar algorithms. It has been observed by Kryściński et al. [4] that many SOTA extraction models only slightly outperform Lead summarization on this dataset, thus the lead model is a benchmark for a relatively good summarizer for CNN/Daily Mail summarization.

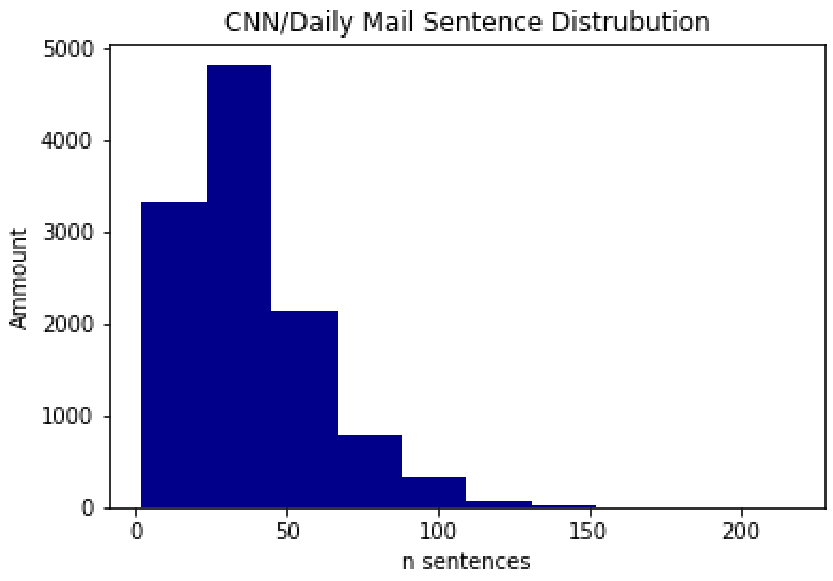
We use the test set from the CNN/Daily Mail dataset, consisting of 11490 pairs of articles and provided summaries. The summaries are abstractive in nature, being provided by manual written summaries for the articles. Therefore, the ROUGE scores for the extractive models should be compared to the Oracle model that provides the greedy-upper bound ROUGE metrics for an extractive summary.

The test-set is filters sentence lengths above 60 to speed up testing. This decreases the test-set to 9907 data points. This might lead to some biased results, as the algorithms aren't tested on larger document summarizations, but was necessary to meet the given time constraints.

Example Data Point

Article
(CNN)Justin Timberlake and Jessica Biel, welcome to parenthood. The celebrity couple announced the arrival of their son, Silas Randall Timberlake, in statements to People. "Silas was the middle name of Timberlake's maternal grandfather Bill Bomar, who died in 2012, while Randall is the musician's own middle name, as well as his father's first," People reports. The couple announced the pregnancy in January, with an Instagram post. It is the first baby for both. Timberlake and Biel welcome son Silas Randall Timberlake .

Highlights
The couple announced the pregnancy in January .


CNN/Daily Mail Sentence Distrubution
Ammount
n sentences
0
1000
2000
3000
4000
5000
0
50
100
150
200

4. Method

The models are all ran on the CNN/Daily Mail, producing a candidate summary for each article, and then computing ROUGE metrics based on the model summary and the gold summary.

We test the models on different amounts of sentences in the summary. This is due to two reasons:

- Some of the algorithms has possible modifications that chooses the optimal number of sentences for a summary, whilst other algorithms does not. For this reason, the algorithms are tested on a fixed amount of sentences, to be able to compare the algorithms properly.
- To study how well algorithms scales with the number of sentences, which could be useful for model choice in practice when a specific amount of sentences is necessary.

4.1 Metrics

The model's summaries are evaluated using F1 ROUGE score. These are defined as:

$$ROUGE_nPrecision = \frac{\mid ngrams_{GoldSummary} \cap ngrams_{ProposedSummary} \mid}{\mid ngrams_{ProposedSummary} \mid} \tag{4.1}$$

$$ROUGE_nRecall = \frac{\mid ngrams_{GoldSummary} \cap ngrams_{ProposedSummary} \mid}{\mid ngrams_{GoldSummary} \mid} \tag{4.2}$$

$$ROUGE_nScore = \frac{2 * ROUGE_nPrecision * ROUGE_nRecall}{ROUGE_nPrecision + ROUGE_nRecall} \tag{4.3}$$

$$ROUGEL - Precision = \frac{\mid LCS(GoldSummary, ProposedSummary) \mid}{\mid ngrams_{ProposedSummary} \mid} \tag{4.4}$$

$$ROUGEL - Recall = \frac{\mid LCS(GoldSummary, ProposedSummary) \mid}{\mid ngrams_{GoldSummary} \mid} \tag{4.5}$$

$$ROUGEL - Score = \frac{2 * ROUGEL - Precision * ROUGEL - Recall}{ROUGEL - Precision + ROUGEL - Recall} \tag{4.6}$$

(LCS : Longest Common Subsequence)

NOTE: summaries are compared to the gold summary which is handwritten, and is therefore in some cases abstractive. As the models produces extractive summarize, the gold and proposed can rarely be equal. This is why the evaluation necessitates baselines that measure an optimistic and pessimistic bound for the ROUGE scores.

4.2 Evaluated Models

All models below uses the simple intuitive assumption that ordering the sentences according to the ordering in the original document should usually lead to a more comprehensive summary. If the document contains less sentences than the current evaluated sentence amount, the entire document is used as a summary. Also, each of the models below are tested using sentence-embeddings with both TFIDF, and with S-BERT embeddings to measure relative performance difference. As previously mentioned, each model is tested through the dataset with $nsentences \in \{1 \dots 4\}$ as the summary length.

4.2.1 Oracle

The Oracle model was proposed by Nallapati et al. [6] to serve as an upper bound for extractive summarization tasks. This model's ROUGE metrics, specifically ROUGE-2, are used as an optimistic bound for the CNN/Daily Mail dataset. The Oracle model greedily maximises the ROUGE-2 Score for each document. This is done iteratively in the following manner: Find the sentence that has not yet been added to the summary from $sentences_{document}$ that when added to the current summary maximizes the ROUGE-2 score for the gold summary. Then add this sentence. Repeat until $nsentences$ are added to the summary.

Note that this algorithm does not guarantee a global optimum. This would necessitate checking all combinations of sentences, leading to a very large amount of computations when multiple sentences are evaluated. The number of computations for the global optimum would thus be $\binom{sentences_{document}}{nsentences}$ leading to a complexity of $O(\mid sentences_{document} \mid^{nsentences})$. This is why the greedy approach is used instead.

4.2.2 Random

This model's ROGE metrics are used as a pessimistic bound for the dataset. It simply chooses $nsentences$ randomly from the document and use this as a summary.

4.2.3 Lead

This model simply picks the first $nsentences$ in the document as the summary. This is derived from the observation by Kryściński et al. [4] that the first three sentences often outperform sophisticated extractive algorithms, especially for datasets of news articles. Therefor, the performance of the Lead model serves as a baseline for a "good" summary.

4.2.4 TextRank

This algorithm produces a sentence similarity matrix M , where each element $M[i,j]$ corresponds to the cosine-similarity measure of the evaluated sentence embedding: $1 - \frac{s_i \cdot s_j}{|s_i| * |s_j|}$. s_i and s_j are encoded and evaluated using TFIDF and the pre-trained S-BERT separately. The M matrix symbolises a graph, where $M[i,j]$ are vertexes between two nodes. ($M[i,i] \forall i \ = 0$) to adhere to the PageRanks assumption of no self-cycles). The PageRank algorithm is then ran for 100 iterations with a damping factor $d = 0.85$:

4.2.5 MatchSum-U

This algorithm uses a greedy summary construction algorithm inspired by the work of Zhong et al. [10]. In this paper a siamese network is trained to minimize the sentence embeddings' distance between the gold summarize and the summary with highest ROUGE score. Instead, we simply use the pre-trained S-BERT sentence embeddings and TFIDF vectors to compute the document-summary similarities iteratively, adding the sentence that maximise the document-summary similarity until *nsentences* are added to the summary. As this summary proposition model is very similar to the one used in MatchSum, we choose to call this model MatchSum-U, for unsupervised MatchSum. The summary embedding is computed as the average of the individual sentence embeddings, and the document embedding is the averaged sentence embeddings for the entire article.

4.2.6 Clustering

Clustering for extractive summarization has been explored amongst others by Agarwal et al. [1] and shows promising results. This algorithm uses KMeans clustering over the sentence-embedding vectors for a document. The amount of clusters k is equal to the number of sentences *nsentences*. The sentence closest to each cluster centre is picked for the summary. Each cluster could in theory correspond to a sentence set that share semantic meaning I.E a sort of "topic", and the sentence closest to the centre of the cluster is most similar to all other sentences in the cluster thus describing the "topic" the most. Intuitively, this algorithm thus picks one sentence from each "topic" in the document that best describes this "topic".

In the works of Argawal et al. [1], the sentence embeddings' dimensions are decreased significantly by the use of PCA to counteract the fact that K-Means performs worse on higher dimensional data. This modifications to the model is worth exploring in future work.

4.2.7 LSA

This algorithm makes use of Singular Value Decomposition as described in section 2.4. SVD is performed on the matrix of sentence embeddings S, where each column $S[,j]$ is the embedding of sentence j. The SVD is ran with "topics" $k = 6$. Running SVD on the matrix S produces the two matrixes U and V with dimensions *embeddingsize x k* and *k x nsentences* respectively, and with the topic importance vector Σ [3]. Then, two variants of this algorithm is tested:

- **None Weighted**: 1. Pick the top *nsentences* most important "topics", by sorting the Σ array descending and picking the first *nsentences* entries. 2. For each topic, find the sentence with

the largest weight for this topic $argmax(V[topic,])$ use this sentence in the summary, remove it from the matrix to prevent duplicates, and continue the algorithm until the summary contains $nsentences$.
Intuitively, step 1 corresponds to finding the topics that describes the document the most, and step 2 corresponds to picking one sentence for each topic that describes that particular topic the most. This approach was suggested by Gong and Liu (2001) [3].

- **Weighted**: Compute the matrix product $\Sigma * V$ and pick the top $nsentences$ entries. This intuitavely is like computing the total topic importance for all sentences. This approach was suggested by Steinberger and Jezek (2004) [9].

Note that $k >= nsentences$ must hold.

4.3 Delimitations

Automatic text summarization is a subset of the larger research field of NLP (Natural Language Processing) and can be divided into two major categories of algorithms:

- Extractive: Summarizes are built by selecting n sentences from the target text.
- Abstractive: Summarizes are generated with any method that can generate sentences not present in the target text.

This work aims to only consider extractive models that does not need labelled data for training, I.E unsupervised extractive text summary models. All models used with Sentence-Embeddings are compared to their equivalent with TFIDF word vectors. Some classes of high performing extractive unsupervised models are ignored such as Integer Linear Programming optimization using bipartite LDA-Sentence graphs, as these does not rely on sentence-sentence similarity measures.

Also, the work only considers sentence embeddings from the S-BERT model, as this model has seen SOTA performance for semantic representation of sentences which should intuitively give good summarization performance, although investigating other word embedding models such as SkipThought embeddings could have lead to a more rigorous result.

There exists opportunities for parameter tuning for some of the algorithms that could lead to increases in performance measures. One issue with parameter tuning is that different datasets can lead to different optimal values. Also, since the task is considered as unsupervised, parameter tuning would not be possible on an unlabelled dataset and would need a reference value for a similar dataset, leading to uncertainty of the parameter value. Since this research is mostly concerned with the relative performance between models and not to achieve the best possible performance on the CNN/Daily Mail dataset, the parameter tuning is not as relevant. The most notable of these potential parameters that could be optimized and could be useful for future research are:

- Minimum similarity threshold for the TextRank algorithm. The PageRank algorithm usually uses a threshold of minimum number of links/similarity to build a link in the graph between two nodes.

- Number of iterations and damping factor for computation in TextRank algorithm.
- Default number of topics in the LSA model.

The LSA model in particular has many alterations possible as proposed in the literature. For instance, one method proposed by Ozsoy et al. is to modify the similarity matrix M by setting entries in the matrix to zero if they are less than the row average. As most of these methods are based on the usage of TFIDF, the modifications are ignored as they have not been considered for Word-Embeddings.

5. Results

The results on running the algorithms on the filtered CNN/Daily Mail test-set consisting of 9907 articles are displayed below. The ROUGE scores are presented for each amount of fixed sentences used in the summary. Table 5.4 displays the relative performance change from using BERT-Embeddings instead of TFIDF, calculated with $\frac{score_{BERT}}{score_{TFIDF}} - 1$ and averaged for all summary lengths. The emphasized scores shows which of the models (not counting the baselines) that performs best.

Table 5.1: ROUGE-1 F1 Scores

Model	1	2	3	4
Oracle	0.411	0.519	0.542	0.456
Random	0.153	0.252	0.286	0.199
Lead	0.260	0.370	0.396	0.288
LSA Weighted-TFIDF	0.186	0.174	0.233	**0.284**
LSA Weighted-BERT	0.083	0.116	0.180	0.233
LSA None Weighted-TFIDF	0.142	0.225	0.253	0.205
LSA None Weighted-BERT	0.134	0.203	0.231	0.192
Cluster-TFIDF	0.174	0.219	0.243	0.180
Cluster-BERT	0.230	0.259	0.272	0.206
MatchSum-U-TFIDF	0.174	0.230	0.248	0.175
MatchSum-U-BERT	0.228	0.276	**0.283**	0.207
TextRank-TFIDF	0.234	0.270	0.279	0.207
TextRank-BERT	**0.237**	**0.278**	0.281	0.197

Table 5.2: ROUGE-2 F1 Scores

Model	1	2	3	4
Oracle	0.262	0.329	0.340	0.277
Random	0.030	0.072	0.085	0.029
Lead	0.090	0.152	0.173	0.114
LSA Weighted-TFIDF	0.062	0.029	0.048	**0.098**
LSA Weighted-BERT	0.020	0.017	0.030	0.068
LSA None Weighted-TFIDF	0.037	0.052	0.055	0.031
LSA None Weighted-BERT	0.030	0.042	0.046	0.024
Cluster-TFIDF	0.054	0.056	0.057	0.028
Cluster-BERT	0.053	0.058	0.060	0.032
MatchSum-U-TFIDF	0.054	0.064	0.064	0.032
MatchSum-U-BERT	0.055	0.063	0.064	0.033
TextRank-TFIDF	**0.070**	**0.072**	**0.072**	0.039
TextRank-BERT	0.059	0.066	0.068	0.035

Table 5.3: ROUGE-L F1 Scores

Model	1	2	3	4
Oracle	0.338	0.393	0.394	0.346
Random	0.110	0.166	0.181	0.122
Lead	0.182	0.234	0.250	0.194
LSA Weighted-TFIDF	0.1396	0.1204	0.1457	**0.1889**
LSA Weighted-BERT	0.0655	0.0836	0.1177	0.1561
LSA None Weighted-TFIDF	0.103	0.148	0.155	0.128
LSA None Weighted-BERT	0.094	0.131	0.143	0.120
Cluster-TFIDF	0.121	0.137	0.143	0.109
Cluster-BERT	0.154	0.160	0.160	0.123
MatchSum-U-TFIDF	0.122	0.143	0.147	0.108
MatchSum-U-BERT	0.155	0.167	0.164	0.125
TextRank-TFIDF	**0.164**	**0.175**	**0.172**	0.131
TextRank-BERT	0.161	0.172	0.167	0.122

Table 5.4: Performance Increases BERT/TFIDF (%)

Model	Rouge-1	Rouge-2	Rouge-L
LSA Weighted	-32	-44	-30
LSA Non Weighted	-8	-19	-8
Cluster	+19	**+5**	+17
MatchSum-U	**+21**	+2	**+18**
TextRank	0	-10	-4

6. Discussion

The best performing model was Weighted LSA with TFIDF at a four sentence long summary.

The models that achieves the largest performance gains from the usage of S-BERT sentence embeddings are MatchSum-U and Cluster. Meanwhile, LSA and TextRank mostly saw a performance decrease from the usage of the S-BERT embeddings. The two methods that gain performance from S-BERT both use similarity measures as their primary method of generating the summary. As S-BERT mainly improves sentence similarity by encoding semantic meaning, this improvement makes intuitive sense, as these models now use a similarity measure based on sentence meaning instead of word distribution. LSA usually uses TFIDF, as the resulting factorization then analyses the actual sentence-word matrix. Using sentence embeddings, the rows in the corresponding matrix no longer refers to actual words but instead the embedding dimensions, which possibly makes the same LSA methods unusable as they no longer refer to concrete word-sentence relationships, possibly explaining the decrease in performance.

It was surprising that TextRank did not see any performance increases from using S-BERT embeddings. The TextRank algorithm's weights, corresponding to sentence similarity, should in theory be more semantically correct with S-BERT embeddings than with TFIDF. This might be due to implementation level detail issues, for instance, the implemented algorithm was found to slightly diverge from the suggested TextRank algorithm after testing was concluded. The main difference was that this algorithm did not divide the adjecency matrix by the sum of columns (which corresponds to the denominator sum in Equation 2.1). Also since the cosine-distance is used, weights in the graph can be negative, and this might be an issue if the weight-sum cancels out to zero. The test scripts were ran for a short time with adjustments for these two issues, but no significant change in performance was observed.

In general, TextRank performs the best out of all models for summary lengths less than four sentences. LSA Weighted with TFIDF performs best in all measures at four sentences long summaries, being very close to the performance of the Lead summaries. Another interesting observation is that None-Weighted LSA performs better than Weighted LSA on all metrics less than four sentence long summaries. This seems to indicate that Weighted LSA scales well with the amount of sentences in the summary, but more experiments with longer summaries are needed to confirm this statement.

Interestingly, the best performing algorithms only outperforms the random summarization when using one sentence or four sentences. The performance of the models could be fine tuned further for better results, for instance the TextRank algorithm's damping factor has not been tuned. Also, the Clustering algorithm suggested by Agarwal et al. [1] was simplified as discussed in subsection 4.2.6, potentially limiting performance. This might also indicate that using a pre-trained summarizer such as Pegasus or MatchSum is preferred, but this approach could be limited on the dissimilarity of the pre-training task on the target dataset as discussed in section 1.2.

Even the best performing models that were tested did not come close to the performance of supervised extractive models on the CNN/Daily Mail dataset. For instance, MatchSum achieves Rouge-1: 0.44 Rouge-2: 0.21 and Rouge-L: 40.55 [10] compared to weighted LSA which achieves Rouge-1: 0.284 Rouge-2: 0.098 and Rouge-L: 0.1889.

7. Conclusion

It appears that significant performance increases can be gained from using sentence embeddings from a pre-trained S-BERT model when using extractive summarization models that utilize sentence similarity measures as their main method for summary generation. Increases in the range of 20% for clustering and document-sentence similarity models has been observed in this work.

The best performing algorithms of those tested for this particular task and dataset, which was TextRank and Weighted LSA, performs better with TFIDF sentence embeddings.

TextRank-TFIDF performed the best out of all models on sentence-summary length of 1-3, whilst LSA Weighted-TFIDF performed the best at sentence-summary length of 4 which indicates that LSA potentially scales better for summaries with larger number of sentences.

The performance difference for the unsupervised models tested and the SOTA supervised models such as MatchSum for the CNN/Daily Mail dataset is significant, by a factor of 2-3 times higher performance in ROUGE measures for MatchSum compared to LSA Weighted-TFIDF [10]. This shows that supervised models are to be preferred when the summarization task has labelled data or is similar enough to an existing dataset.

Bibliography

[1] S. Agarwal, N. K. Singh, and P. Meel, *Single-document summarization using sentence embeddings and k-means clustering*, 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 162–165.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[3] Yihong Gong and Xin Liu, *Generic text summarization using relevance measure and latent semantic analysis*, 01 2001, pp. 19–25.

[4] Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher, *Neural text summarization: A critical evaluation*, 2019.

[5] Rada Mihalcea and Paul Tarau, *TextRank: Bringing order into text*, Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (Barcelona, Spain), Association for Computational Linguistics, July 2004, pp. 404–411.

[6] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou, *Summarunner: A recurrent neural network based sequence model for extractive summarization of documents*, CoRR **abs/1611.04230** (2016).

[7] Makbule Ozsoy, Ferda Alpaslan, and Ilyas Cicekli, *Text summarization using latent semantic analysis*, J. Information Science **37** (2011), 405–417.

[8] Nils Reimers and Iryna Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019.

[9] Josef Steinberger and Karel Jezek, *Using latent semantic analysis in text summarization and summary evaluation*, 01 2004.

[10] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang, *Extractive summarization as text matching*, 04 2020.