Sparse Optimization for Unsupervised Extractive Summarization of Long Documents with the Frank-Wolfe Algorithm

Alicia Y. Tsai
University of California, Berkeley
aliciatsai@berkeley.edu

Laurent El Ghaoui
University of California, Berkeley
elghaoui@berkeley.edu

Abstract

We address the problem of unsupervised extractive document summarization, especially for long documents. We model the unsupervised problem as a sparse auto-regression one and approximate the resulting combinatorial problem via a convex, norm-constrained problem. We solve it using a dedicated Frank-Wolfe algorithm. To generate a summary with k sentences, the algorithm only needs to execute $\approx k$ iterations, making it very efficient. We explain how to avoid explicit calculation of the full gradient and how to include sentence embedding information. We evaluate our approach against two other unsupervised methods using both lexical (standard) ROUGE scores, as well as semantic (embedding-based) ones. Our method achieves better results with both datasets and works especially well when combined with embeddings for highly paraphrased summaries.

1 Introduction

With the overwhelming increase of digital information, automatic text summarization has become important for many applications such as financial reviews, medical articles, etc. Manually summarizing this amount of information takes a considerable amount of time and effort. This has motivated the study of efficient and reliable automatic text summarization methods. The task of automatic summarization is the process of generating a condensed version of a text that best describes the original one (Hahn and Mani, 2000; Luhn, 1958). The two mainstream approaches in the field of automatic summarization are *extractive* and *abstractive*.

Extractive approaches generate summaries by selecting a subset of informative words, phrases, or sentences directly from the source text. In contrast, abstractive approaches use linguistic methods to decompose and build a semantic representation of the text and use natural language generation techniques to generate a summary (Chopra et al., 2016; Nallapati et al., 2016; Zeng et al., 2016; Rush et al., 2015). In recent years, neural network architectures have made abstractive summarization popular. However, abstractive approaches are generally harder to develop as they require high performing natural language generation techniques, which is also an active research field. Besides these two categories, mixed strategies that combine both extractive and abstractive approaches have also been proposed in recent literature (Peng et al., 2019; Cao et al., 2018; See et al., 2017). Previous work in extractive approaches to summarization include statistical (Saggion and Poibeau, 2012; Das and Martins, 2007; Goldstein et al., 1999; Kupiec et al., 1995; Paice, 1990), graph-based and optimization-based ones. The graph-based approaches treat text as a network instead of as a simple bag of words and use graph-based ranking methods to generate a summary (Erkan and Radev, 2011; Ouyang et al., 2009; Mihalcea and Tarau, 2004). Optimization-based methods use techniques such as sparse optimization (Yao et al., 2015; Elhamifar and Vidal, 2013), integer linear programming (ILP) (Qian and Liu, 2013; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2011; Gillick and Favre, 2009) and constraint optimization (Durrett et al., 2016; McDonald, 2007) to reconstruct the summary.

In this work, we focus on extractive summarization for long documents. Performing automatic text summarization for long documents is especially challenging as obtaining high quality human summaries for long documents is often quite costly and time consuming. Recent works on extractive summarization have been focusing on neural network architectures (Nallapati et al., 2017; Cheng and Lapata, 2016). Although these methods are successful in generating summaries for short documents, they often have difficulties with long input

Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 54–62
Online, November 20, 2020. ©2020 Association for Computational Linguistics

sequences (Shao et al., 2017).

Most recent works have started to investigate neural extractive summarization methods for long documents (Xiao and Carenini, 2019; Wang et al., 2017). However, these methods are supervised and require high quality training data in order to train the neural network models. This creates challenges for domains that do not have massive training datasets. Kedzie et al. (2018) compared recent neural extractive summarization models across different domains including news, personal stories, and medical articles. They found that many sophisticated neural extractive summarizers do not have better performance than those consisting of simpler models, and that word embedding averaging performs equally or better than RNNs or CNNs for sentence embedding. This suggests that a simpler model combined with pre-trained word embeddings show promise for summarizing long documents in domains that have few or no training data.

In this work, we propose an unsupervised method for extracting long documents based on a sparse optimization framework and solve it using a dedicated Frank-Wolfe algorithm, which can be combined with pre-trained word embeddings to construct a distributive input representation. Our work is based on the previous work of Cheng et al. (2018) but designed specifically for the summarization task. The proposed framework is an unsupervised model that is efficient and does not require a training corpora, as typical supervised solutions would require. We test our method on two datasets that contain long documents, 2019 FINANCIAL OUTLOOKS and CLASSICAL LITERATURE, and compare it against two baselines: sparse subspace clustering (SSC) and TextRank. The experimental results show that our method gives a higher ROUGE score than our baseline for both datasets. In particular, when combined with sentence embedding, our method gives a higher semantic ROUGE score when evaluated on paraphrased summaries. Moreover, we show that our method is computationally more efficient compared to others.

2 Methodologies

Notation We denote $X_{(i)}$ and $X^{(i)}$ as the i-th row and i-th column of a matrix X respectively. The matrix X_t denotes the value of X at iteration t while $[X_t]_{(i)}$ and $[X_t]^{(i)}$ denotes the i-th row and i-th column of X_t. The sum $\sum_{i=1}^{n} \|X_{(i)}\|_2$ is the L_2 norm group LASSO constraint. The norm $\|\cdot\|_F$ is the Forbenium norm.

2.1 Sparse auto-regressive problem

Extractive summarization aims at finding a minimal set of representative sentences of the original document that effectively summarizes the entire document. Let $A \in \mathbb{R}^{d\times n}$ be the data matrix that represents the document where each column of A represents a sentence in the source document. Here, d is the number of features for each sentence, and n is the number of sentences in the source document. The source document A is written as

$$A \triangleq \begin{bmatrix} a_1 & a_2 \cdots & a_n \end{bmatrix}$$

where the column vector a_i is a sentence in the source document. Finding the set of representative sentences assumes that the source document A can be approximated by a sparse combination of sentences in the document:

$$A \approx a_1 x_1^T + a_2 x_2^T + \cdots + a_n x_n^T$$

The column vector x_i is a decision variable to be learned. Our goal is to select k sentences whose corresponding decision variable x_i is non-zero. If we write it in a matrix form with x_i^T being the row of the matrix variable X, we can formulate the above problem as an auto-regressive problem of the form:

$$\begin{aligned} \min_X \quad & \|AX - A\|_F^2 \\ \text{s.t.} \quad & \|v\|_0 \le k \\ & v_i = \|X_{(i)}\|_2, \quad \forall i \in [n] \\ & X \ge 0 \end{aligned} \tag{1}$$

where X is row-sparse. Here, $X_{(i)}$ represents the i-th row of the matrix variable X, v_i is its norm, and the constraint is written in terms of the L_0-norm (cardinality, or number of non-zero entries) of v, effectively forcing at least $n - k$ entire rows of X to be zero, thereby singling out a short list of at most k sentences that well represent the whole data set.

The above problem is non-convex and hard to solve but can be well approximated by the so-called L_1-norm heuristic, leading to a convex approximation:

$$\begin{aligned} \min_X \quad & \|AX - A\|_F^2 \\ \text{s.t.} \quad & \|v\|_1 \le \beta \\ & v_i = \|X_{(i)}\|_2, \quad \forall i \in [n] \\ & X \ge 0 \end{aligned} \tag{2}$$

where β is a hyper-parameter and indirectly controls the row-sparsity (number of non-zero rows). Note that the model simply uses the L_1-norm of vector v in (1) to approximate the cardinality constraint on v. If X^* is the solution of problem (2), then columns in the data matrix $A^{(j)}$ that correspond to the non-zero rows of X^*, $X_{(j)} \neq 0$, are the selected sentences.

2.2 Frank-Wolfe unsupervised extractive summarization

The Frank-Wolfe (FW) or conditional gradient algorithm is an iterative first-order optimization algorithm for constrained convex optimization (Frank and Wolfe, 1956). Although the algorithm was introduced over half a century ago, it has experienced a revival in recent years due to its projection-free iterations and broad applications in machine learning (Jaggi, 2013).

The FW algorithm solves a general constrained optimization problem of the form $\min_{x\in\mathcal{D}} f(x)$, where the convex function f is differentiable and L-Lipschitz and the domain $\mathcal{D}$ is a convex compact set. At each iteration, the FW algorithm requires solving a linear approximation to the objective function over the domain, often referred to as a *linear minimization oracle* (LMO), and then updates the solution accordingly. At each iteration, we first calculate the gradient, solve the LMO problem to find a descent direction, calculate the step size by line search or by $\frac{2}{t+2}$, and update the estimate. Algorithm 1 summarizes the FW process.

Algorithm 1 Frank-Wolfe algorithm

```
1: Let $t \leftarrow 0$ and $\boldsymbol{x}_0 \in \mathcal{D}$
2: for $t = 0, 1, \dots,$ do
3:     $\boldsymbol{s}_t = \arg\min_{\boldsymbol{s}\in\mathcal{D}} \langle \boldsymbol{s}, \nabla f(\boldsymbol{x}_t)\rangle$
4:     Set step size $\boldsymbol{r}_t \leftarrow \frac{2}{t+2}$ or
5:     $\boldsymbol{r}_t \leftarrow \arg\min_{\boldsymbol{r}\in[0,1]} f(\boldsymbol{x}_t + \boldsymbol{r}(\boldsymbol{s}_t - \boldsymbol{x}_t))$
6:     Update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{r}_t(\boldsymbol{s}_t - \boldsymbol{x}_t)$
7: return $\boldsymbol{x}_t$
```

Unlike other descent methods for constrained optimization that require a projection step at each iteration, the FW algorithm is a projection-free algorithm and only needs to solve the LMO. Applying the FW algorithm to our sparse constrained optimization problem (2) results in an unsupervised method for extractive summarization. Algorithm 1 is written in terms of vector variable x; however, it is straightforward to extend it to the matrix variable X. The algorithm starts with $X_0 \leftarrow 0$, meaning no sentence is selected at first. Then the algorithm greedily selects one sentence at each iteration. The algorithm terminates once k rows of X_t are non-zero (dense) or when the algorithm converges to a row-sparse solution with $k^* < k$ non-zero rows. The complete algorithm of Frank-Wolfe unsupervised extractive summarization is outlined in algorithm 2. Next, we explain the details of the algorithm and provide an efficient gradient calculation scheme.

Linear minimization oracle The algorithm requires solving the LMO at each iteration. The solution of the LMO S_t specifies the direction of descent at each step.

$$S_t = \arg\min_{S\in\mathcal{D}} \langle S, \nabla f(X_t)\rangle$$

Because of the group LASSO constraint in (2), the solution matrix S_t is a rank-1 matrix. The non-zero row of S_t is chosen based on the maximum L_2 norm of the gradient's rows:

$$j = \arg\max_{i} = \|[\nabla f(X_t)]_{(i)}\|_2$$

We denote the non-zero row of S_t at row j as $[S_t]_{(j)}$. The magnitude of $[S_t]_{(j)}$ is β and the direction is chosen to minimize the inner product:

$$[S_t]_{(j)} = -\beta \frac{[\nabla f(X_t)]_{(j)}}{\|[\nabla f(X_t)]_{(j)}\|_2}$$

The algorithm produces sparse and low-rank iterates since at most one extra row of X becomes non-zero in each step by the addition of $\boldsymbol{r}_t S_t$.

Efficient gradient calculation The algorithm requires calculating the gradient at each iteration. The gradient of the objective function in (2) is:

$$\nabla f(X) = 2(A^TAX - A^TA) = 2(KX - K)$$

The matrix $K = A^TA$ can be calculated once and used throughout the algorithm. Explicitly calculating the gradient is expensive due to the matrix-matrix product (naively $\mathcal{O}(n^3)$). However, the structure of the problem allows us to efficiently calculate KX at each iteration. From line 6 in Algorithm 1, we know that X_t is a weighted average of X_{t-1} and a rank-1 matrix S_{t-1}:

$$X_t = (1 - \boldsymbol{r}_{t-1})X_{t-1} + \boldsymbol{r}_{t-1}S_{t-1}$$

This suggests that $(KX)_t$ is a weighted average of $(KX)_{t-1}$ and $KS_{t-1} = K^{(j)}[S_{t-1}]_{(j)}$. $K^{(j)}$ is the j-th column corresponding to the j-th non-zero row of S_{t-1}. Since $(KX)_{t-1}$ is known at iteration t, we are only required to calculate $K^{(j)}[S_{t-1}]_{(j)}$, which is extremely fast (in $\mathcal{O}(n)$).

Stopping criteria The algorithm terminates once k rows of X_t are non-zero (dense) or when X_t converges to a row-sparse solution such that $-\langle \nabla f(X_t), S_t - X_t \rangle < \epsilon$. Once the algorithm terminates, it returns the k sentences that correspond to the non-zero rows of X_t by **GetSummary**(X_t, k).

Sentence similarity measure We note that the gradient of (2) depends only on the kernel (or, "Gram") matrix $K = A^T A$ and not on A. This matrix is akin to a similarity matrix, with K_{ij} measuring the similarity between sentences i and j. If the matrix A is normalized, K_{ij}'s are cosine similarities. As a result, we may replace the matrix K with any matrix $\Phi(A)$ that offers a good similarity measure between sentences. This allow us to incorporate various sentence scoring functions $\Phi(\cdot)$. In this paper, we experimented with two such similarity measures: 1) TF-IDF-like, and 2) sentence embedding.

For TF-IDF-like similarity measure, we use Okapi BM25 (Robertson and Zaragoza, 2009) to construct the kernel matrix K. BM25 and its variants represent the state-of-the-art TF-IDF-like sentence scoring functions. Similarly, any sentence embedding technique can be used to embed the document matrix A in a much lower dimensional space; that is, we can set $K_{ij} = \phi(a_i)^T \phi(a_j)$, with $\phi(a)$ the (low-dimensional) vector representing the sentence a. In this work, we use a simple yet effective sentence embedding method called smooth inverse frequency (SIF) (Arora et al., 2017) to measure the sentence similarities. In Arora et al. (2017), the authors show that SIF, a simple weighted average of word vectors modified by SVD, outperforms complex methods such as RNNs and LSTMs. More sophisticated sentence embedding techniques such as neural architectures can also be used here; however, once should also consider the cost of computing the kernel matrix K with such a technique.

In the following, the acronyms FWSum-BM25 and FWSum-SIF are used to refer to the corresponding Frank-Wolfe unsupervised extractive summarization method used in conjunction with the BM25 and SIF similarity kernels.

3 Experiments

3.1 Datasets

Dernoncourt et al. (2018) surveyed the current large-scale dataset for summarization. Most of them are relatively short; usually less than 2 pages. To experiment on long documents, we used the recently open-sourced 2019 FINANCIAL OUTLOOKS and CLASSICAL LITERATURE dataset[1], which contain much longer documents than those surveyed in Dernoncourt et al. (2018).

2019 FINANCIAL OUTLOOKS This corpus contains 10 publicly available reports on finance from a number of large financial institutions. Each report ranges from 10 to 144 pages, with a median length of 33 pages. There are no Gold summaries *per se* since the data is not annotated by a human. Hence, we chose to define the gold summaries as the collection of sentences or parts of sentences that appear in bold in the content, or any sentences that are highlighted as an insert within the content. This is a reasonable heuristic as these parts are generally prepared by the authors to highlight the takeaway of the content.

CLASSICAL LITERATURE The corpus contains summaries of books that have been summarized by human writers. The corpus contains 11 English-language classical books ranging from 53 to 1139 pages, with a median length of 198 pages. The Gold summaries for each chapter of the book are retrieved from WikiSummary [2].

3.2 Baselines

We compared our method with two other unsupervised extractive approaches; one uses a sparse optimization-based method and the other uses a graph-based method.

Sparse subspace clustering (SSC) Sparse subspace clustering (SSC) solves a sparse optimization program on the auto-regressive problem similar to (1), called the *self-expressiveness property* of the data (Elhamifar and Vidal, 2013). This property assumes that each data point can be efficiently reconstructed by a combination of other points in the data and that there exists a sparse representation

[1] https://github.com/SumUpAnalytics/goldsum

[2] http://wikisum.com/w/Main_Page

Algorithm 2 Frank-Wolfe unsupervised extractive summarization

```
input $\beta, k, \epsilon$
initialize $X_0, (KX)_0, \boldsymbol{r}_0, t \leftarrow 0, 0, 0, 1$
compute $K = A^T A$ or $\Phi(A)$
for $t = 1, 2, \ldots$ do
    $(KX)_t = (1 - \boldsymbol{r}_{t-1})(KX)_{t-1} + \boldsymbol{r}_{t-1} K^{(j)} [S_{t-1}]_{(j)}$
    $\nabla f(X_t) = 2\big((KX)_t - K\big)$
    $j = \arg\max_j \big\| [\nabla f(X_t)]_{(j)} \big\|_2$
    $S_t = 0$
    $\big[S_t\big]_{(j)} = -\beta \frac{\big[\nabla f(X_t)\big]_{(j)}}{\big\| \big[\nabla f(X_t)\big]_{(j)} \big\|_2}$
    $\boldsymbol{r}_t = \frac{2}{t+2}$ or $\arg\min_{\boldsymbol{r} \in [0,1]} f(X_t + \boldsymbol{r}(S_t - X_t))$
    $X_{t+1} = X_t + \boldsymbol{r}_t (S_t - X_t)$
    if NumSent$(X_{t+1}) = k$ or $-\big\langle \nabla f(X_t), S_t - X_t \big\rangle < \epsilon$ then
        break                                  ▷ $k$ rows are non-zero or $X_t$ converges
    $t = t + 1$
return GetSummary$(X_{t+1}, k)$
```

of the data point. The authors consider a convex relaxation as we did in (2) since solving the original sparse optimization is in general NP-hard. SSC uses the Alternating Direction Method of Multipliers (ADMM) for solving the sparse optimization problem. In our work, we employ the Frank-Wolfe algorithm on the problem, which is more efficient compared to SSC. Subsequent work tried to speed up SSC (You et al., 2016) but with an expense of removing the group LASSO constraint that is crucial for our summarization problem. In our work, we are able to preserve the group LASSO constraint and obtain a faster run-time. In our experiment, we used the implementation of Elhamifar and Vidal (2013), which can be found on their website[3].

TextRank TextRank (Mihalcea and Tarau, 2004) is a commonly used graph-based unsupervised extractive summarization method. It is also very efficient when extracting summaries from a long document. TextRank employs the similar idea of PageRank where vertices in the graph are sentences in the document and edges between two sentences are measured as a function of their content overlap.

3.3 Lexical and semantic ROUGE scores

We evaluate the systems using the ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) so as to account for different summary lengths. The raw ROUGE score only measures the lexical overlaps between the generated summaries and the reference summaries. We refer to the raw ROUGE score defined in Lin (2004) as the *lexical ROUGE* and used the implementation of the Python `rouge` library[4]. When summarizing a long document, humans tend to paraphrase the source document in order to condense and synthesize the information. However, the lexical ROUGE scores are unable to measure the quality of paraphrasing. To address this shortcoming of lexical ROUGE when the summaries are paraphrased, word embedding ROUGE scores (Ng and Abrecht, 2015) are also used to evaluate the quality of the generated summaries. The word embedding ROUGE scores are more capable of measuring semantic similarity of the words instead of only lexical overlaps. Ng and Abrecht (2015) showed that the embedding ROUGE achieved better correlations with human assessments compared to lexical ROUGE when measured with the Spearman and Kendall rank coefficients on the TAC AESOP summarization dataset. We refer to the word embedding ROUGE scores as the *semantic ROUGE* in our evaluation.

4 Results and Analysis

In our experiment, we set the number of selected sentences k to be the same as the length of reference summary for all methods. The performance of

[3] http://www.ccs.neu.edu/home/eelhami/codes.htm

[4] https://pypi.org/project/rouge/

Lexical ROUGE		SSC	TextRank	FWSum-BM25	FWSum-SIF
Financial Outlook	ROUGE-L F1	11.88	14.43	13.92	**14.99**
	ROUGE-2 F1	2.15	3.76	**5.17**	3.05
	ROUGE-1 F1	14.6	**19.94**	**19.9**	18
Classical Literature	ROUGE-L F1	7.48	16.27	**18.7**	13.18
	ROUGE-2 F1	0.38	2.54	**3.23**	1.25
	ROUGE-1 F1	9.97	19.61	**20.2**	16.58

Semantic ROUGE		SSC	TextRank	FWSum-BM25	FWSum-SIF
Financial Outlook	ROUGE-L F1	30.01	26.2	22.97	**34.56**
	ROUGE-2 F1	55.56	61.43	**61.82**	58.92
	ROUGE-1 F1	43.4	48.28	**49.97**	47.73
Classical Literature	ROUGE-L F1	31.92	39.15	39.1	**46.6**
	ROUGE-2 F1	47.73	53.35	54.1	**60.53**
	ROUGE-1 F1	38.72	44.15	42.43	**48.18**

Table 1: Lexical and semantic ROUGE performance for Financial Outlook and Classical Literature data. Results that are statistically better are bold faced and results that are statistically indistinguishable are colored as gray. An additional experimental results can be found in appendix A.

all methods on Financial Outlook and Classical Literature are shown in Table 1. As shown in the table, FWSum-BM25 has a similar performance with TextRank although slightly better. This may be explained by the sentence scoring functions used by TextRank and FWSum-BM25. TextRank uses lexical overlaps between two sentences while FWSum-BM25 uses the TF-IDF-like scoring function, which are similar in nature.

FWSum-BW25 performs especially well when evaluated with lexical ROUGE, highlighting its capabilities of capturing lexical information (measured by unigram and bigram). When evaluated on the Financial Outlook data, FWSum-BW25 and TextRank generally outperform FWSum-SIF, with FWSum-BM25 being the best performing method. Presumably, this is due to the fact that the Gold summaries of the Financial Outlook data are taken directly from the source document without much paraphrasing, favoring sentence scoring functions that directly measure the content overlaps.

However, when evaluated by the semantic ROUGE on the Classical Literature data, FWSum-SIF start to show promises. The Gold summaries of the Classical Literature data are written by human writers and are highly paraphrased and condensed. As a result, semantic ROUGE is a better measurement for this dataset. As shown in the table, FWSum-SIF starts to outperform other methods by a significant amount. The improvement over the other methods suggests that using embedding in the sentence scoring function allows for comparisons based on the semantics of words sequences.

This results show that different sentence scoring functions may be used based on the nature of the summary. For summaries that are mostly taken from the source document without much paraphrasing, a lexical overlap or TF-IDF-like kernel matrix may be used. For summaries that are highly paraphrased, an embedding-like kernel matrix may be more suitable. Our method is able to work with both.

Computational complexity Our method requires an up-front cost of calculating the kernel matrix K. Each subsequent iteration requires mostly the LMO and gradient calculation as detailed in section 2. By exploiting the structure of the problem, we are able to avoid explicitly calculating the full gradient. Furthermore, due to the greedy nature of the algorithm, it terminates when k sentences are selected or the solution converges with $k^* < k$ sentences. This means that the algorithm only needs to execute $\approx k$ iterations; each iteration has a cost linear in problem size. Figure 1 compares the algorithm run-time of our method (FWSum-BM25), TextRank and SSC. As shown in the figure, our method is the most efficient among the three, showing its potential for summarizing long documents.

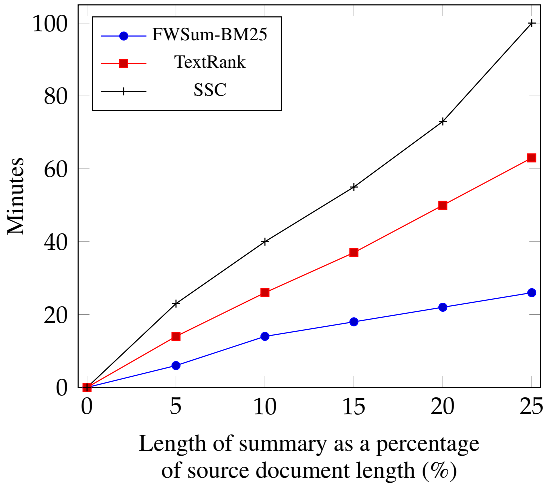

Figure 1: Algorithm run-time for FWSum-BM25, TextRank and SSC on the FINANACIAL OUTLOOK data. The x-axis shows the length of the generated summary (*i.e.* k) as a percentage of the source document length (number of sentences in the source document).

5 Conclusion

Unsupervised document summarization has been a challenging task, especially on long documents. In this work, we propose an efficient unsupervised extractive summarization model that is suitable for long documents by employing a dedicated Frank-Wolfe algorithm. Our method allows one to incorporate sentence embedding or any sentence scoring functions that is best suited for the dataset or the application. We evaluate our method and compare it with two other unsupervised extractive summarization methods on two datasets that are much longer than other summarization corpora used in the past. We evaluate the methods on both lexical and semantic ROUGE in order to overcome the shortcoming of lexical ROUGE and to provide a better assessment of the quality of the summaries. We observed that our methods (both FWSum-BM25 and FWSum-SIF) achieve the best results for both datasets and that FWSum-SIF works especially well with summaries that are paraphrased. Our results also motivate the exploration of different kernel functions or embedding methods, which is left as a future work.

Acknowledgments

The authors would like to thank Richard Liou, Tanya Roosta, and Gary Cheng for their constructive discussions on drafts of this paper and SumUp Analytics for providing the dataset.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 481–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Gary Cheng, Armin Askari, Laurent El Ghaoui, and Kannan Ramchandran. 2018. Frank-wolfe algorithm for exemplar selection. *CoRR*, abs/1811.02702.

Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Dipanjan Das and André F. T. Martins. 2007. A survey on automatic text summarization.

Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. 2018. A Repository of Corpora for Summarization. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *CoRR*, abs/1603.08887.

Ehsan Elhamifar and Rene Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *arXiv:1203.1005 [cs, math, stat]*. ArXiv: 1203.1005.

Günes Erkan and Dragomir R. Radev. 2011. Lexrank: Graph-based lexical centrality as salience in text summarization. *CoRR*, abs/1109.2128.

Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, Boulder, Colorado. Association for Computational Linguistics.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 121–128, New York, NY, USA. ACM.

Udo Hahn and Inderjeet Mani. 2000. The challenges of automatic summarization. *Computer*, 33(11):29–36.

Martin Jaggi. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435.

Chris Kedzie, Kathleen McKeown, and Hal Daumé III. 2018. Content Selection in Deep Learning Models of Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium. Association for Computational Linguistics.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73, New York, NY, USA. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: a recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3075–3081, San Francisco, California, USA. AAAI Press.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930, Lisbon, Portugal. Association for Computational Linguistics.

You Ouyang, Wenjie Li, Furu Wei, and Qin Lu. 2009. Learning similarity functions in graph-based document summarization. In *Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, pages 189–200, Berlin, Heidelberg. Springer Berlin Heidelberg.

C. D. Paice. 1990. Constructing literature abstracts by computer: Techniques and prospects. *Inf. Process. Manage.*, 26(1):171–186.

Hao Peng, Ankur P. Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. *CoRR*, abs/1904.04428.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502, Seattle, Washington, USA. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Horacio Saggion and Thierry Poibeau. 2012. Automatic Text Summarization: Past, Present and Future. In R. Yangarber T. Poibeau; H. Saggion. J. Piskorski, editor, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 3–13. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models. *arXiv:1701.03185 [cs]*. ArXiv: 1701.03185.

Shuai Wang, Xiang Zhao, Bo Li, Bin Ge, and Daquan Tang. 2017. Integrating Extractive and Abstractive Models for Long Text Summarization. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 305–312.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Wen Xiao and Giuseppe Carenini. 2019. Extractive Summarization of Long Documents by Combining Global and Local Context. *arXiv:1909.08089 [cs]*. ArXiv: 1909.08089.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1376–1382. AAAI Press.

Chong You, Daniel P. Robinson, and Rene Vidal. 2016. Scalable Sparse Subspace Clustering by Orthogonal Matching Pursuit. *arXiv:1507.01238 [cs, stat]*. ArXiv: 1507.01238.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *CoRR*, abs/1611.03382.