Extractive Summarization by Maximizing Semantic Volume

Dani Yogatama
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dyogatama@cs.cmu.edu

Fei Liu
Electrical Engineering & Computer Science
University of Central Florida
Orlando, FL 32816, USA
feiliu@cs.ucf.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

The most successful approaches to extractive text summarization seek to maximize bigram coverage subject to a budget constraint. In this work, we propose instead to maximize semantic volume. We embed each sentence in a semantic space and construct a summary by choosing a subset of sentences whose convex hull maximizes volume in that space. We provide a greedy algorithm based on the Gram-Schmidt process to efficiently perform volume maximization. Our method outperforms the state-of-the-art summarization approaches on benchmark datasets.

1 Introduction

In artificial intelligence, changes in representation sometimes suggest new algorithms. For example, increased attention to distributed meaning representations suggests that existing combinatorial algorithms for NLP might be supplanted by alternatives designed specifically for embeddings. In this work, we consider summarization.

Classical approaches to extractive summarization represent each sentence as a bag of terms (typically bigrams) and seek a subset of sentences from the input document(s) that either (a) trade off between high relevance and low redundancy (Carbonell and Goldstein, 1998; McDonald, 2007), or (b) maximize bigram coverage (Yih et al., 2007; Gillick et al., 2008). The sentence representation is fundamentally *discrete*, and a range of greedy (Carbonell and Goldstein, 1998), approximate (Almeida and Martins, 2013), and exact optimization algorithms (McDonald, 2007; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011) have been proposed.

Recent studies have explored *continuous* sentence representations, including the paragraph vector (Le and Mikolov, 2014), a convolutional neural network architecture (Kalchbrenner et al., 2014), and a dictionary learning approach (Jenatton et al., 2011). If sentences are represented as low-dimensional embeddings in a distributed semantic space, then we begin to imagine a geometric relationship between a summary and a document. We propose that the **volume** of a summary (i.e., the semantic subspace spanned by the selected sentences) should ideally be large. We therefore formalize a new objective function for summarization based on semantic volume (§2), and we provide a fast greedy algorithm that can be used to maximize it (§3). We show that our method outperforms competing extractive baselines under similar experimental conditions on benchmark summarization datasets (§4).

2 Extractive Summarization Models

Assume we are given a set of N sentences: $\mathcal{D} = \{s_1, s_2, \ldots, s_N\}$ from one or many documents, and the goal is to produce a summary by choosing a subset $\mathcal{S}$ of M sentences, where $\mathcal{S} \subseteq \mathcal{D}$ and $M \leq N$, and the length of the summary is less than or equal to L words. In this work, we assume no summaries are available as training data. Denote a binary indicator vector $\mathbf{y} \in \mathbb{R}^N$, where sentence i is included if and only if $y_i = 1$ and 0 otherwise. Extractive summarization can be written as an optimization problem:

$$\begin{array}{lr} \text{max} & \text{score}(\mathcal{S}) = \text{score}(\mathcal{D}, \mathbf{y}) \\ \text{with respect to} & \mathcal{S} \text{ equivalently } \mathbf{y} \\ \text{subject to} & \text{length}(\mathcal{S}) \leq L \end{array}$$

with a scoring function score$(\mathcal{D}, \mathbf{y})$. A good scoring function should assign higher scores to better summaries. In the following, we describe two commonly used scoring functions and our proposed scoring function.

2.1 Maximal Marginal Relevance

The Maximal Marginal Relevance (MMR) method (Carbonell and Goldstein, 1998) considers the following scoring function:

$$\text{score}(\mathcal{D}, \mathbf{y}) = \sum_{i=1}^{N} y_i \text{Rel}(s_i) - \sum_{i,j=1}^{N} y_i y_j \text{Sim}(s_i, s_j)$$

where Rel(s_i) measures the relevancy of sentence i and Sim(s_i, s_j) measures the (e.g., cosine) similarity between sentence i and sentence j. The intuition is to choose sentences that are highly relevant to the document(s) and avoid redundancy. The above maximization problem has been shown to be NP-hard, solvable exactly using ILP (McDonald, 2007). A greedy algorithm that approximates the global solution by adding one sentence at a time to maximize the overall score (Lin and Bilmes, 2010) is often used in practice.

2.2 Coverage-Based Summarization

Another popular scoring function aims to give higher scores for covering more diverse concepts in the summary. Gillick et al. (2008) use bigrams as a surrogate for concepts. Following convention, we extract bigrams from each sentence $s_i \in \mathcal{D}$. Denote the number of unique bigrams extracted from all sentences by B. We introduce another binary vector $\mathbf{z} \in \mathbb{R}^B$ to indicate the presence or absence of a bigram in the summary, and a binary indicator matrix $\mathbf{M} \in \mathbb{R}^{N \times B}$, where $m_{i,j}$ is 1 if and only if bigram j is present in sentence i and 0 otherwise. The scoring function is:

$$\text{score}(\mathcal{D}, \mathbf{y}, \mathbf{z}) = \sum_{j=1}^{B} b_j z_j$$

and the two additional constraints are:

$$\forall j \in [B], \forall i \in [N] \qquad y_i m_{i,j} \leq z_j$$

$$\forall j \in [B] \qquad \sum_{i=1}^{N} y_i m_{i,j} \geq z_j$$

where we use $[B]$ as a shorthand for $\{1, 2, \ldots, B\}$. The first constraint makes sure that selecting a sentence implies selecting all its bigrams, whereas the second constraint makes sure that selecting a bigram implies selecting at least one of the sentences that contains it. In this formulation, there is no explicit penalty on redundancy. However, insofar as redundant sentences cover fewer bigrams, they are implicitly discouraged. Although the above scoring function also results in an NP-hard problem, an off-the-shelf ILP solver (Gillick et al., 2008) or a dual decomposition algorithm (Almeida and Martins, 2013) can be used to solve it in practice.

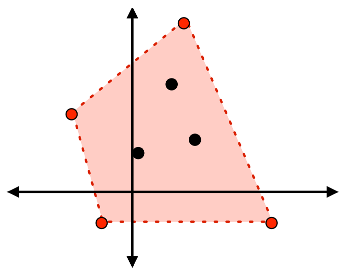
Figure 1: A toy example of seven sentences projected into a two-dimensional semantic space. Consider the case when the maximum summary length is four sentences. Our scoring function is optimized by chooseing the four sentences in red as the summary, since they maximize the volume (area in two dimensions).

2.3 Semantic Volume

We introduce a new scoring function for summarization. The main idea is based on the notion of coverage, but in a distributed semantic space: a good summary should have broad semantic coverage with respect to document contents. For every sentence $s_i, i \in [N]$, we denote its continuous semantic representation in a K-dimensional semantic space by $\Omega(s_i) = \mathbf{u}_i \in \mathbb{R}^K$, where Ω is a function that takes a sentence and returns its semantic vector representation. We denote embeddings of all sentences in $\mathcal{D}$ with the function Ω by $\Omega(\mathcal{D})$. We will return to the choice of Ω later. We propose to use a scoring function that maximizes the *volume* of selected sentences in this semantic space:

$$\text{score}(\mathcal{D}, \mathbf{y}) = \text{Volume}(\Omega(\mathcal{D}), \mathbf{y}) = \text{Volume}(\Omega(\mathcal{S}))$$

In the case when $K = 2$, this scoring function maximizes the area of a polytope, as illustrated in Figure 1. In the example, there exists a maximum number of sentences that can be selected such that adding more sentences does not increase the score, i.e., the set of selected sentences forms a convex hull of the set of all sentences. The sentences forming a convex hull may together be longer than

L words, so we seek to maximize the volume of the summary under this constraint.

There are many choices of Ω that we can use to produce sentence embeddings. As an exploratory study, we construct a vector of bigrams for each sentence, that is, $\mathbf{s}_i \in \mathbb{R}^B, \forall i \in [N]$. If bigram b is present in s_i, we let $s_{i,b}$ be the number of documents in the corpus that contain bigram b, and zero otherwise. We stack these vectors in columns to produce a matrix $\mathbf{S} \in \mathbb{R}^{N \times B}$, where N is the number of sentences in the corpus and B is the number of bigrams. We then perform singular value decomposition (SVD) on $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. We use $\mathbf{U}_K \in \mathbb{R}^{N \times K}$ as the sentence representations, where K is a parameter that specifies the number of latent dimensions. Instead of performing SVD, we can also take $\mathbf{s}_i \in \mathbb{R}^B$ as our sentence representation, which makes our method resemble the bigram coverage-based summarization approach. However, this makes $\mathbf{s}_i$ a very sparse vector. Projecting to a lower dimensional space makes sense to allow the representation to incorporate information from (bigram) cooccurrences and share information across bigrams.

3 Volume Maximization

Given the semantic coverage scoring function in §2.3, our optimization problem is:

$$\begin{array}{ll} \max & \text{score}(\mathcal{S}) = \text{Volume}(\Omega(\mathcal{S})) \\ \text{with respect to} & \mathcal{S} \\ \text{subject to} & \text{length}(\mathcal{S}) \leq L \end{array}$$

For computational considerations, we propose to use a greedy algorithm that approximates the solution by iteratively adding a sentence that maximizes the current semantic coverage, given that the length constraint is still satisfied. The main steps in our algorithm are as follows. We first find the sentence that is farthest from the cluster centroid and add it to $\mathcal{S}$. Next, we find the sentence that is farthest from the first sentence and add it to $\mathcal{S}$. Given a set of already selected sentences, we choose the next one by finding the sentence farthest from the subspace spanned by sentences already in the set. We repeat this process until we have gone through all sentences, breaking ties arbitrarily and checking whether adding a sentence to $\mathcal{S}$ will result in a violation of the length constraint. This method is summarized in Algorithm 1. We note that related variants of our method for maximizing volume have appeared in

Algorithm 1 Greedy algorithm for approximately maximizing the semantic volume given a budget constraint.

```
Input: Budget constraint $L$, sentence representations $\mathcal{R} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\}$
  $\mathcal{S} = \{\}, \mathcal{B} = \{\}$
  Compute the cluster centroid $\mathbf{c}$: $\frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_i$.
  $p \leftarrow$ index of sentence that is farthest from $\mathbf{c}$.
  $\mathcal{S} = \mathcal{S} \cup \{s_p\}$.                ▶ add first sentence
  $q \leftarrow$ index of sentence that is farthest from $\mathbf{s}_p$.
  $\mathcal{S} = \mathcal{S} \cup \{s_q\}$.                ▶ add second sentence
  $\mathbf{b}_0 = \frac{\mathbf{u}_q}{\|\mathbf{u}_q\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{u}_0\}$
  total length = length$(s_p)$ + length$(s_q)$
  for $i = 1, \ldots, N-2$ do
    $r \leftarrow$ index of sentence that is farthest from
    the subspace of Span$(\mathcal{B})$.                ▶ see text
    if total length + length$(s_r) \leq L$ then
      $\mathcal{S} = \mathcal{S} \cup \{s_r\}$.
      $\mathbf{b}_r = \frac{\mathbf{u}_r}{\|\mathbf{u}_r\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{b}_r\}$.
      total length = total length + length$(s_r)$
    end if
  end for
```

other applications, such as remote sensing (Nascimento and Dias, 2005; Gomez et al., 2007) and topic modeling (Arora et al., 2012; Arora et al., 2013).

Computing Distance to a Subspace Our algorithm involves finding a point farthest from a subspace (except for the first and second sentences, which can be selected by computing pointwise distances). In order for this algorithm to be efficient, we need this operation to be fast, since it is executed frequently. There are several established methods to compute the distance between a point to a subspace spanned by sentences in $\mathcal{S}$. For completeness, we describe one method based on the Gram-Schmidt process (Laplace, 1812) here.

We maintain a set of basis vectors, denoted by $\mathcal{B}$. Our first basis vector consists of one element: $\mathbf{b}_0 = \frac{\mathbf{u}_q}{\|\mathbf{u}_q\|}$, where q is the second sentence chosen above. Next, we project each candidate sentence i to this basis vector:

$$\text{Proj}_{\mathbf{b}_0}(\mathbf{u}_i) = (\mathbf{u}_i^\top \mathbf{b}_0)\mathbf{b}_0,$$

and find the distance by computing Distance$(\mathbf{u}_i, \mathcal{B}) = \|\mathbf{u}_i - \text{Proj}_{\mathbf{b}_0}(\mathbf{u}_i)\|$. Once we find the farthest sentence r, we add a new basis vector $\mathcal{B} = \mathcal{B} \cup \{\mathbf{b}_r\}$, where $\mathbf{b}_r = \frac{\mathbf{u}_r}{\|\mathbf{u}_r\|}$ and repeat this process. When there are more than one basis vectors, we find the distance by computing:

$$\text{Distance}(\mathbf{u}_i, \mathcal{B}) = \left\| \mathbf{u}_i - \sum_{\mathbf{b}_j \in \mathcal{B}} \text{Proj}_{\mathbf{b}_j}(\mathbf{u}_i) \right\|.$$

4 Experiments

4.1 Setup

We evaluate our proposed method on the non-update portion of TAC-2008 and TAC-2009. The datasets contain 48 and 44 multi-document summarization problems, respectively. Each problem has 10 news articles as input; each is to be summarized in a maximum of $L = 100$ words. There are 4 human reference summaries for each problem, against which an automatically generated summary is compared. We compare our method with two baselines: Maximal Marginal Relevance (MMR, §2.1) and the coverage-based summarization method (CBS, §2.2). ROUGE (Lin, 2004) is used to evaluate the summarization results.

For preprocessing, we tokenize, stem with the Porter (1980) stemmer, and split documents into sentences. We remove bigrams consisting of only stopwords and bigrams which appear in less than 3 sentences. As a result, we have 2,746 and 3,273 bigrams for the TAC-2008 and TAC-2009 datasets respectively. Unlabeled data can help generate better sentence representations. For each summarization problem in each dataset, we use other problems in the same dataset as unlabeled data. We concatenate every problem in each dataset and perform SVD on this matrix (§2.3). Note that this also means we only need to do one SVD for each dataset.

4.2 Results

Table 1 shows results on the TAC-2008 and TAC-2009 datasets. We report results for our method with $K = 500$ (Volume 500), and $K = 600$ (Volume 600). We also include results for an oracle model that has access to the human reference summaries and extracts sentences that maximize bigram recall as an upper bound. Similar to previous findings, CBS is generally better than MMR. Our method outperforms other competing methods, although the optimal value of K is different in each dataset. The improvements with our proposed approach are small in terms of R-2. This is likely because the R-2 score computes bigram overlaps, and the CBS method that directly maximizes bigram coverage is already a resonable approach to optimizing this metric (although still worse than the best of our methods).

Methods	TAC-2008		TAC-2009	
	R-1	R-2	R-1	R-2
MMR	34.08	9.30	31.87	7.99
CBS	35.83	9.43	32.70	8.84
Volume 500	37.40	9.17	34.08	**8.91**
Volume 600	**37.50**	**9.58**	**34.37**	8.76
Oracle	46.06	19.33	46.77	16.99

Table 1: Results on the TAC-2008 and TAC-2009 datasets. "Volume" refers to our method, shown with two embedding sizes.

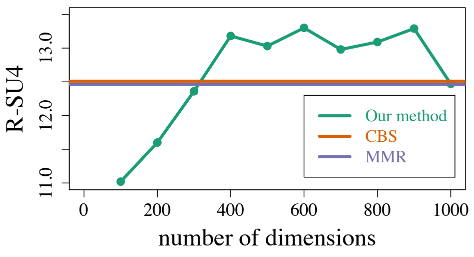

Figure 2: R-SU4 scores as we vary the number of dimensions (K) on the TAC-2008 datasets.

5 Discussion

Runtime comparisons In terms of inference running time, all methods perform reasonably fast. MMR is the slowest, on average it takes 0.38 seconds per problem, followed by our method at 0.17 seconds per problem, and CBS at 0.15 seconds per problem. However, our implementations of MMR and Algorithm 1 are in Python, whereas we use an optimzed solver from Gurobi for our CBS baseline. For preprocessing, our method is the slowest, since we need to compute sentence embeddings using SVD. There are about 10,000 sentences and 3,000 bigrams for each dataset. SVD takes approximately 2.5 minutes (150 seconds) using Matlab on our 12-core machine with 24GB RAM. Our method introduces another hyperparameter, the number of latent dimensions K for sentence embeddings. We observe that the optimal value depends on the dataset, although a value in the range of 400 to 800 seems best. Figure 2 shows R-SU4 scores on the TAC-2008 dataset as we vary K.

Other sentence projection methods We use SVD in this study for computing sentence embeddings. As mentioned previously, our summarization approach can benefit from advances in neural-network-based sentence representations (Jenatton et al., 2011; Le and Mikolov, 2014; Kalchbrenner et al., 2014). These models can also produce vector representations of sentences, so Algorithm 1 can be readily applied to the learned representations. Our work opens up a possibility to make summarization a future benchmark task for evaluating the quality of sentence representations.

Our method is related to determinantal point processes (DPPs; Gillenwater et al., 2012; Kulesza and Taskar, 2012) in that they both seek to maximize the volume spanned by sentence vectors to produce a summary. In DPP-based approaches, quality and selectional diversity correspond to vector magnitude and angle respectively. In this work, the length of a sentence vector is not tailored to encode quality in terms of representativeness directly. In contrast, we rely on sentence embedding methods to produce a semantic space and assume that a good summary should have a large volume in the semantic space. We show that a simple singular value decomposition embedding of sentences—one that is not especially tuned for this task—produces reasonably good results. We leave exploration of other sentence embedding methods to future work.

Future work Our method could be extended for compressive summarization, by simply including compressed sentences in the embedded space and running Algorithm 1 without any change. This resembles the summarization methods that jointly extracts and compresses (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013). Another alternative is a pipeline approach, where extractive summarization is followed or preceded by a sentence compression module, which can be built and tuned independent of our proposed extractive method (Knight and Marcu, 2000; Lin, 2003; Zajic et al., 2007; Wang et al., 2013; Li et al., 2013).

We are also interested in exploring volume as a relevance function within MMR. MMR avoids redundancy by penalizing redundant sentences, whereas in our method semantic redundancy is inherently discouraged since the method chooses sentences to maximize volume. Depending on the method used to embed sentences, this might not translate directly into avoiding n-gram redundancy. Plugging our scoring function to an MMR objective is a simple way to enforce diversity.

Finally, an interesting future direction is finding an exact tractable solution to the volume maximization problem (or demonstrating that one does not exist).

6 Conclusion

We introduced a summarization approach based on maximizing volume in a semantic vector space. We showed an algorithm to efficiently perform volume maximization in this semantic space. We demonstrated that our method outperforms existing state-of-the-art extractive methods on benchmark summarization datasets.

Acknowledgments

We thank anonymous reviewers for helpful suggestions. This work was supported by the Defense Advanced Research Projects Agency through grant FA87501420244 and by NSF grant SaTC-1330596. This work was completed while the authors were at CMU.

References

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of ACL*.

Sanjeev Arora, Rong Ge, Ravi Kannan, and Ankur Moitra. 2012. Computing a nonnegative matrix factorization – provably. In *Proc. of STOC*.

Sanjeev Arora, Rong Ge, Yoni Halpren, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proc. of ICML*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proc. of ACL*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. 2012. Discovering diverse and salient threads in document collections. In *Proc. of EMNLP-CoNLL*.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The ICSI summarization system at TAC 2008. In *Proc. of TAC*.

C. Gomez, H. Le Borgne, P. Allemand, C. Delacourt, and P. Ledru. 2007. N-findr method versus independent component analysis for lithological identification in hyperspectral imagery. *International Journal of Remote Sensing*, 28(23):5315–5338.

Rodolphe Jenatton, Julien Mairal, Gullaume Obozinski, and Francis Bach. 2011. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. of ACL*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. In *Proc. of AAAI*.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286.

Pierre-Simon Laplace. 1812. *Theorie analytique des probabilites*. Courcier, Paris.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of ICML*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proc. of EMNLP*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proc. of NAACL-HLT*.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression – A pilot study. In *Proc. of Workshop on Information Retrieval with Asian Language*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proc. of the ACL Workshop on Text Summarization Branches Out*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proc. of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proc. of ECIR*.

Jose M. P. Nascimento and Jose M. Bioucas Dias. 2005. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *Proc. of IEEE Transaction on Geoscience and Remote Sensing*, 43(4):898–910.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Lu Wang, Hema Raghavan Vittorio Castelli Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multidocument summarization. In *Proc. of ACL*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proc. of EMNLP*.

Wen-Tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proc. of IJCAI*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570.