SummVD : An efficient approach for unsupervised topic-based text summarization

Gabriel Shenouda[1] **Christophe Rodrigues**[1] **Aurélien Bossard**[2]
(1) Léonard De Vinci Pôle Universitaire, Research Center, 92 916 Paris La Défense, France
(2) Laboratoire d'Informatique Avancée de Saint-Denis, Université Paris 8 (EA4383)
93200 Saint-Denis, France

Abstract

This paper introduces a new method, SummVD, for automatic unsupervised extractive summarization. This method is based on singular value decomposition, a linear method in the number of words, in order to reduce the dimensionality of word embeddings and propose a representation of words on a small number of dimensions, each representing a hidden topic. It also uses word clustering to reduce the vocabulary size. This representation, specific to one document, reduces the noise brought by several dimensions of the embeddings that are useless in a restricted context. It is followed by a linear sentence extraction heuristic. This makes SummVD an efficient method for text summarization. We evaluate SummVD using several corpora of different nature (news, scientific articles, social network). Our method outperforms in effectiveness recent extractive approaches. Moreover, SummVD requires low resources, in terms of data and computing power. So it can be run on long single documents such as scientific papers as much as large multi-document corpora and is fast enough to be used in live summarization systems.

1 Introduction

Research on automatic summarization has recently focused on supervised approaches. Since Pointer Generator by See et al. (2017), there has been considerable advances in the supervised generative summarization field (Zhang et al., 2020; Wu et al., 2021; Liu et al., 2021; Zhong et al., 2020). However, these approaches need substantial learning corpora composed of a large amount of documents and summary pairs, and despite recent advances on fine-tuning and transfer learning, are limited to specific domains. Thus research on unsupervised summarization methods cannot be left out. In this paper, we tackle the problem of unsupervised extractive summarization, which aims to select sentences from one or multiple documents and put them together in order to build a summary. This extraction is often based on centrality and diversity notions : how much is a sentence central to the input text, and how many of the central information is present in the output summary.

Inspired by the work of (Gong et al., 2018) on long texts similarity computation, we assume that hidden topics specific to a text can emerge from word embeddings computed from a general corpus. Each topic stands for a particular aspect of the text semantics. These hidden topics allow to remove unnecessary information from word representations and can be viewed as a new representation of the text. Words can be matched against a hidden topic, and this way, we can derive word centrality scores from a text, originally represented as a word embeddings matrix. Given these word scores, a sentence extraction heuristic can be applied to generate an extractive summary.

We propose a new efficient method for unsupervised extractive summarization, called SummVD, whose code is available online[1]. We present recent unsupervised methods in Section 2. After, we describe our method in Section 3.1. Section 4 presents our experiments led on a large variety of summarization corpora combining single and multi-document benchmarks, in order to test its generalization. The results shown in Section 5 outperform recent unsupervised methods on most of the evaluation corpora, and get sometimes close to supervised methods. We then discuss in Section 6 complexity and scalability of our method. SummVD's ability to run on long and multi-documents makes it an efficient method to summarize any kind of document, like scientific articles.

[1] https://github.com/SummVD/SummVD

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 501–511
November 20–23, 2022.

2 Related work

2.1 Extractive summarization

Extractive summarization is studied since the late 1950's (Luhn, 1958). Symbolic (Edmundson, 1969) as well as semantic (Barzilay et al., 1999) or statistical (Radev et al., 2000) methods have been successfully used for automatic extractive summarization. Linear integer programming (Gillick and Favre, 2009) and evolutionary algorithms (Bossard and Rodrigues, 2017) have also been adapted to extractive summarization.

TextRank (Mihalcea and Tarau, 2004) is summarization method widely used as a baseline. It is a graph-based method that extracts sentences based on the centrality of their words in a graph representation of the document.

To the best of our knowledge, (Padmakumar and He, 2021) is one of the most recent unsupervised extractive summarizer. In an empirical study, it outperforms state-of-the-art approaches on different kinds of texts (news, medical, discussions). The model is similar to the query likelihood model described in (Manning et al., 2008) for information retrieval where a language model is used to estimate the probability of a document given a query. Here, the query is replaced by a candidate sentence for extraction in the summary. So, in a greedy process, sentences are added to the output summary according to the language model probability estimation. The language model used in (Padmakumar and He, 2021) is GPT-2. It is fine-tuned on each dataset in order to get the best results. All of their hyper-parameters are tuned on 200 randomly sampled document-summary pairs, in order to optimize the ROUGE F1 measure. It includes the coefficient of relevance and redundancy from their sentence scoring equation and the number of sentences to select for all extractive methods.

SummPip (Zhao et al., 2020) is a graph compression based unsupervised multi-document summarization method . It converts documents into a sentence graph where nodes are the sentences, and edges are constructed based on lexical chains, discourse level markers, exogen semantic information (WordNet), named entity reference and a simple semantic similarity based on word embedding vectors. It allows them to take into account the linguistic and deep neural representation of the documents. In order to get a k sentences summary, a Laplacian matrix is created based on the sentence graph representation of their document, and compute the first k eigenvectors from that matrix. This way, each sentence has a feature vector. Finally, a k-means clustering method is used to separate those sentences into k clusters. This method is called spectral clustering. The final step consists in multi-sentence compression, which generates single document summaries from clusters. SummPip uses a more evolved version of the shortest path algorithm to select the final sentences used to generate the output summary. A Word2Vec (Mikolov et al., 2013) model fine-tuned on each dataset is used for the embedding part.

Singular Value Decomposition (SVD) on texts was originally used for document comparison in Latent Semantic Analysis (LSA) technique introduced by (Deerwester et al., 1990). Documents are represented with a document-term matrix filled with the occurrences of terms in documents, one term by row and one document by column. So SVD is employed to reduce the number of terms while preserving the similarity between documents. Gong and Liu (2001) were the first to use LSA for automatic summarization. LSA allows to detect the main topics, then the sentences closest to the topics are extracted to constitute a summary.

The method was improved in 2004 by Steinberger and Jezek (2004) by weighting the sentence selection probability by the importance of the topics (proportional to their variance).

2.2 Text representation

GloVe (Pennington et al., 2014) stands for global vectors for word representation. This embedding technique is essentially a log-bilinear model with a weighted least-squares objective. The model is based on the idea that the simple observation of the ratios of word-word co-occurrence probabilities can emphasize a form of meaning. It combines the features of two model families, namely the global matrix factorization and local context window methods. The resulting representations show linear substructures of the vectoring space. The model creation is unsupervised. It was developed at Stanford, and is an open source project.

Recently released, BERT –Bidirectional Encoder Representations from Transformers– is a method of pre-training language representations created by (Devlin et al., 2019). It provides subwords embeddings and sentence representations. It is designed to pre-train bidirectional representations from unlabeled text by jointly conditioning

on both left and right context in all layers. It is used in a large variety of tasks, like question answering, language inference, text and sentence classification, next sentence prediction, text summarization and more.

2.3 Singular Value Decomposition

A Singular Value decomposition (SVD) of a matrix M of size ($m \times n$) is defined as follows:

$$M = U \cdot \Sigma \cdot V^T$$

3 Our Method: SummVD

3.1 Model proposed

Word embeddings provide a vector representation of words based on their context. However, in a specific context, eg a document or several documents about a same topic, most of the information carried by a word embedding is useless and only brings noise to potential semantic computation over it. Even computing semantic similarity between two words using their word embedding is still a challenge (Farouk, 2018). We propose to adapt unsupervised methods in order to exploit these dense vectors and identify the most important sentences of texts. We can represent the texts in a matrix where a row represents a word and a column represents a dimension of the embedding:

Matrix = #Word x #Dimension

Since a summary can be interpreted as a compression of a text, we will compress this matrix. We describe a two step process where we can first reduce the number of words (rows) by a clustering method and then the number of dimensions (columns) by a singular value decomposition. An overview of the model is given at Figure 1.

3.2 Word clustering

In order to reduce the number of words, and thus word vectors, we use an unsupervised vector clustering method. This way, the closest vectorized words supposed to share the same contexts will be grouped in the same cluster. Depending on the clustering method, it is possible to control the number of clusters. Thus, the lower the number of clusters, the higher the compression rate. The words grouped within a cluster will then all be substituted by a unique vector, representing the cluster. The selected vector is chosen as the closest to the centroid, considering all the vectors sharing the same cluster.

With U and V two orthogonal matrix. The matrix U is composed of n orthonormalized eigenvectors associated with the n largest eigenvalues of MM^T. The matrix V is composed of the orthonormalized eigenvectors of M^TM. Σ is a diagonal matrix composed of singular values defined as the non-negative square roots of the eigenvalues of M^TM in a descending order. So considering the first k dimensions ($k < n$) gives us a dimension reduction of the Matrix M which can be used as an approximation.

We propose to use the SVD to reduce the number of dimensions of the word embeddings. Indeed, since the embeddings have a large dimension (300 in our experiments), the SVD has the ability to identify the dimensions carrying most of the information, thus allowing us to keep the most important ones. As in LSA (Deerwester et al., 1990), we name eigenvectors as topics.

3.3 Scoring words

The score of a word given a topic (found by the SVD) is defined by:

$$WordScore(w, t_i) = \frac{\overrightarrow{w} \cdot \overrightarrow{t_i}}{\|\overrightarrow{w}\|} \quad (1)$$

Where $\overrightarrow{w}$ is the vector embedding of the word w and t_i is a topic found by the SVD. The score is a cosine similarity between the word embedding and the topic. Intuitively, the closer a word is to a topic, the more it explains the variation of this axis, therefore the more information it contains and should be selected to be part of the summary.

3.4 Extracting sentences

Here we describe the method to extract the best sentences according to the reduced matrix achieved by clustering and decomposition.

The heuristic described in Algorithm 1 supposes that the first topics found by SVD can be used to extract one representative sentence per topic.

More precisely, to extract one sentence per topic, as described on Algorithm 1, the best sentence of each topic is selected according to the sum of the score of their words normalized by the length of the sentence. So, the closest sentence of the topic is added to the summary. The operation is repeated for each topic. For k sentences in the output summary, the first k topics are used.

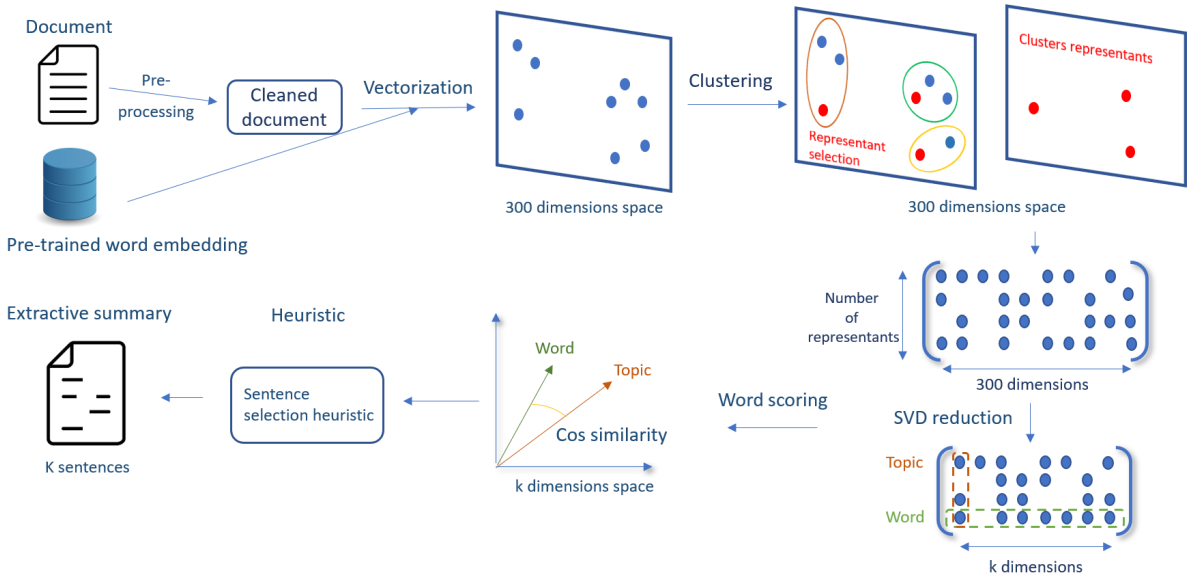

Figure 1: SummVD Pipeline illustrating the sequence of operations needed to achieve an extractive summary from a given text document.

Algorithm 1 SentenceByTopic(D,k)

Require: document D, #sentences k
Ensure: summary sum
$sum = \emptyset$
for all k topics **do**

$$c = \underset{s}{\operatorname{argmax}}^{s \in D} \frac{1}{|s|} \sum_{w}^{w \in S} WordScore(w, k)$$

$$sum = sum \cup c$$

end for

4 Experiment

4.1 Corpora

In order to evaluate our work, we run the evaluation on heterogeneous corpora. For that purpose we compare our method to the two most recent extractive summarization approaches to our knowledge, both on single and multi-document summarization tasks : PMI (Padmakumar and He, 2021) and SummPip (Zhao et al., 2020). Table 1 gives a synthetic view on those corpora features.

CNN/Daily Mail Introduced by (Hermann et al., 2015) for question answering purpose and first used for automatic summarization by (Nallapati et al., 2016). This corpus is composed of newspaper articles extracted from CNN and Daily Mail. Each article is associated to a summary built by concatenating the article highlights defined by its author. Its large scale makes it possible to use in neuronal generative summarization methods. The version we use is the non-anonymized one.

XSum Extreme Summarization dataset (XSum) has been introduced by (Narayan et al., 2018) to evaluate single document summarization systems. Articles are collected from BBC articles (2010 to 2017). Each article is associated to a single sentence summary, more precisely the introductory sentence that prefaces it, professionally written by the author of the article.

PubMed Introduced in (Cohan et al., 2018), it is a single document dataset mainly composed of medical scientific papers associated with their abstract. It consists of long documents.

Reddit Is a Reddit based dataset built by (Ouyang et al., 2017) composed of 476 personal narratives that are used as source documents for summarization. These stories come from 19 different topics and are associated to two gold summaries: an abstractive and an extractive summary, both hand written by four graduate students. We use the same test set as in (Padmakumar and He, 2021), 48 randomly selected examples.

Multi-News Is a multi-document news summarization dataset introduced by (Fabbri et al., 2019). News are extracted from this site[2]. As the majority of text summarization methods use the truncated

[2] http://www.newser.com

Name	Doc nature	type	Test size	sents/doc	words/doc	sents/abst	words/abst	Comp rate
CNN/DM	News	SDS	11489	26.9	766.6	3.9	58.2	7.6%
XSum	News	SDS	11331	23.2	424.9	1	18.6	4.4%
PubMed	Scien paper	SDS	6658	101.6	3142.9	7.6	208	6.6%
Reddit	Soc media	SDS	48	12.1	234.5	1.2	25.2	10.7%
Multi-News	News	MDS	5622	17.5	491	9.8	262.0	53.4%
DUC2004	News	MDS	50	264.9	6583.14	31.12	422.26	6.4%

Table 1: Corpora features: size of test sample (in documents), average number of sentences per document, average number of words per document, average number of sentences and words per abstract (gold standard summaries), and compression rate (cf Equation 2) for each corpus described in Section 4.1

.

version of the corpus, we followed this trend.

DUC 2004 Built for the Document Understanding Conference summarization evaluation campaign, DUC2004 (Over and Liggett, 2004) is a multi-document dataset, which consists of 50 clusters of 10 news articles, each cluster talking about a specific topic. Each of these 50 clusters is paired with a human written summary. Every cluster is concatenated into one document, resulting in a corpus of 50 very long documents, each associated with a gold standard summary.

4.2 Baselines

TextRank We implement TextRank which is a very common and widely spread method across text summarization. This method, described in Section 2, is to this date, one of the quickest unsupervised method to produce summaries. We use the Gensim[3] implementation (Barrios et al., 2016).

LSA We run LSA (Steinberger and Jezek, 2004), a method based on SVD as described in Section 2. It allows to highlight the benefits of our approach using word embeddings.

BERT SVD We implement a completely new approach based on BERT embeddings. It allows to represent not words but entire sentences. Once all the sentences of a document are vectorized, the process is similar as our main approach SummVD. Also the final step of sentence selection is straight, the sentences closest to topics are considered as the best ones.

PMI We run PMI (Padmakumar and He, 2021) using the implementation given by the authors[4]. Our run only concerns single document summarization datasets as PMI is a single document summarization method.

SummPip We run SummPip (Zhao et al., 2020) using the implementation given by the authors[5]. As SummPip is designed for multi-document summarization, our run only concerns multi-document datasets.

Supervised is the MatchSum model (Zhong et al., 2020). It is one of the most recent supervised deep learning extractive approaches.

4.3 Implementation details

We pre-processed the data using the NLTK[6] tools, by eliminating stop words and special characters. We also use the NLTK sentence parser to separate the sentences from the documents.

To achieve a straight comparison between unsupervised text summarization competitors and our approach, we generate summaries of same length as PMI (Padmakumar and He, 2021) and SummPip (Zhao et al., 2020) (in number of sentences). For CNN/DM and XSum we use 3 sentences, for Reddit we use 4 sentences, for PubMed and Multi-News it is 9 sentences, and for DUC 2004, 7 sentences.

In order to keep the method light and truly unsupervised, we empirically decided to use a generic word embedding method: GloVe (Common Crawl, 840B tokens, 2.2M vocab, cased, 300d vectors) which appeared to get the best results.

We tested three clustering methods: OPTICS (Ankerst et al., 1999); an improved version of DBSCAN (Ester et al., 1996), the K-means algorithm (Forgy, 1965), and *Agglomerative Clustering*, all three in their implementation of the scikit-learn library (Pedregosa et al., 2011). The use of *Agglomerative Clustering* induces a slight loss of ROUGE score, of the order of 0.5% to 1.3% compared to k-means and of the order of 1.0% to 1.9% compared to OPTICS, but allows gains in execution

[3] https://radimrehurek.com/gensim/

[4] https://github.com/vishakhpk/mi-unsup-summ

[5] https://github.com/mingzi151/SummPip

[6] https://www.nltk.org/

speed of respectively 40% to 700% and 1100% to 2300% depending on the corpus. The algorithm *Agglomerative Clustering* is thus a good compromise between effectiveness and execution time, an important aspect for the scaling up allowed by the method.

Regarding the number of clusters, we use the elbow method that allows us to find on average and automatically, the number of clusters adapted for each corpus.

Table 1 shows the characteristics of all the corpora described in this section and used in our evaluation process. It highlights the discrepancy between the corpora, in terms of types (single vs multi-document summarization), nature of documents (scientific, newspaper, social media feeds), document and gold standard abstract lengths, and compression rate, given by the following Equation:

$$CompRate(D, A) = \frac{|A|}{|D|} \quad (2)$$

Where D is the source document and A the abstract.

5 Results

In order to evaluate our method, we use the common known ROUGE F1 measure (Lin, 2004). The python library that we use can be found here[7]. This is equivalent to calling the perl ROUGE script as: "ROUGE-1.5.5.pl -m -e ./data -n 2 -a /tmp/rouge/settings.xml".

5.1 ROUGE scores

Table 3 presents our results, using ROUGE F1 scoring. We can see that SummVD outperforms PMI, SummPip and TextRank in most cases. Our method is not always the best but is as effective on single-document than on multi-document summarization tasks, and does not seem to be affected by the document length, which is important for scientific paper summarization or any multi-document summarization task. On both multi-document corpora we tested, our method outperform the others unsupervised methods.

One can see in Table 3 that the supervised method MatchSum heavily outperforms every unsupervised method on the corpora that share a common characteristic: small source documents. However, when it comes to corpora with bigger documents (PubMed and Multi-News) the gap between MatchSum and unsupervised methods tends to decrease.

It is important to note that, considering ROUGE-2, SummVD ranks in first place of unsupervised systems on 5 out of 6 corpora. Graham (2015) has shown that ROUGE-2 is the ROUGE metric that is the most correlated to human evaluation, ROUGE-1 and ROUGE-L being worse ROUGE metrics along with ROUGE-W.

5.2 Execution time

In Table 2, we compare the execution time of SummVD against TextRank, PMI and SummPip. In order to calculate the execution time of PMI and SummPip we do not take into consideration the fine-tuning process of their language model that they actually do on every dataset and that is time consuming. We follow the instructions given on the methods GitHub page, and run the code one by one on a clear work space[8].

We take 500 random examples for each dataset (the same examples for each of the four methods) and run the different methods, measuring the execution time to compute the average time needed to summarize a document.

The first thing to notice is that TextRank is the best performing of all four. It is, in average, 5 times faster than our method. TextRank is well known for being a very quick algorithm, and the Gensim version that we use is optimized to run even faster.

Looking at Tables 1 and 2, one can see that the execution time of SummPip is multiplied by 141 when the number of words per document is multiplied by 6.74 (Multi-News vs DUC2004) when SummVD execution time is only multiplied by 2.2. As a result, our method SummVD is 1626 times quicker in average than SummPip on DUC2004.

Comparing our method to PMI shows that we are in average 885 times quicker on the 5 datasets on which both PMI and SummVD are ran.

There is in average, 6.74 times more words in PubMed than in CNN/DM, XSum, and Reddit. In average, our method execution time is 4.28 times longer on PubMed than on the other 4 datasets. In comparison, PMI has a 8.62 times ratio. Finally PMI is 1494 times slower than our method on PubMed.

To put in perspective, the supervised state-of-the-art baseline MatchSum (Zhong et al., 2020) needs

[7] https://pypi.org/project/rouge-score/

[8] The machine used to perform the calculations has an AMD 3700X 8 cores processor, 64 GB of RAM, and 2 RTX 2080TI of 11GB of memory each and runs on Windows 10

	Mono-document				Multi-document	
	CNN/DM	Xsum	Reddit	PubMed	Multi-News	DUC2004
TextRank	0.02s	0.01s	0.01s	0.09s	0.046s	0.32s
PMI	72.72s	56.28s	25s	448.2s	-	-
SummPip	-	-	-	-	6s	846s
SummVD	0.1s	0.07s	0.05s	0.3s	0.23s	0.52s

Table 2: Average summarization time of every method described in section 4.2 on every corpus described in Table 1 for one document.

TextRank: a father-of-three and popular radio host in berkeley, california, was killed in a hit-and-run in the early hours of saturday morning. Wesley burton, a father-of-three and popular radio host at kpfa in berkeley, california, was killed in a hit-and-run in the early hours of saturday morning as he drove home from work. burton had three children - santiago, enrique, and samaya – aged between 4 and 9 and after growing up without a father his dream had been to raise his own kids

LSA: the crash occurred near the berkeley-oakland city line and police say the hit-and-run driver fled on foot. a gofundme account has been set up to help burton 's wife pay funeral costs and other family expenses. police are urging anyone with information to call the traffic investigation unit on (510)777-8570.

PMI: a father-of-three and popular radio host in berkeley, california, was killed in a hit-and-run in the early hours of saturday morning. his wife lucrecia has made a tearful plea for anyone with information to come forward and speak to the police. we lost our rock. he was our stability, our strength, ' she told ktvu.

BERT SVD: ' help us regain our peace. burton had three children - santiago, enrique, and samaya – aged between 4 and 9. oakland crime stoppers is offering a $ 10,000 reward for information leading to an arrest.

SummVD: a father-of-three and popular radio host in berkeley, california, was killed in a hit-and-run in the early hours of saturday morning. wesley burton, who worked at kpfa, was driving home from work when a white dodge charger crashed into his silver mercury. the crash occurred near the berkeley-oakland city line and police say the hit-and-run driver fled on foot.

Figure 2: Examples of summaries generated by SummVD and different baselines exposed in §4.2 on a same article belonging to CNN/DM corpus.

30 hours just for training only for the CNN/DM corpus on an heavy dedicated machine (8 GPUs V100).

6 Discussion

6.1 Complexity

To the best of our knowledge, apart from MMR (Carbonell and Goldstein, 1998) and its derivate methods, there is no fully linear method to generate extractive summaries. The complexity of the SVD (Golub and Van Loan, 1996) is defined by:

$$O(mn\ min\{n, m\})$$

In our case, m the number of words and n the size of the word embedding.

An interesting point is that in your specific case the number of columns is fixed by the size of the embedding (here 300) but remains unchanged independently of the document size. So, increasing the size of documents will only add new lines (words). As a result, for documents with a number of words superior than the size of the embedding, the SVD complexity is quadatric in n and linear in m. Since n is fixed, the complexity of the SVD becomes linear in number of words when $m > 300$.

It's explains why your approach scale well when number of words increases. This theoretical result opens the possibility to process large documents in practice, as shown in Figure 3.

6.2 Scalability

The complexity of SummVD, illustrated in Figure 3 on a logarithmic scale allows us to scale up. The comparison against the gensim (Rehurek and So-

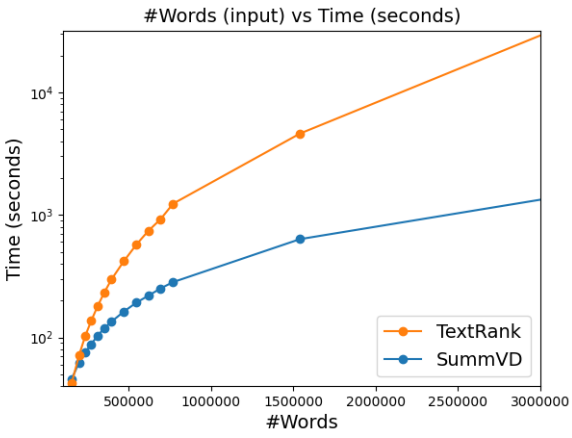

Figure 3: Average time to compute a summary, against the number of input words for SummVD and TextRank (gensim implementation). Time is in logarithmic scale.

	Mono-document												Multi-document					
	CNN/DM			XSum			Reddit			PubMed			Multi-News			DUC2004		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Supervised	44.41	20.86	-	24.86	04.66	-	-	-	-	41.21	14.91	-	46.20	16.51	-	-	-	-
Lead-k	**40.13**	17.63	**25.09**	19.52	02.67	12.45	25.66	07.51	17.94	37.98	13.55	20.16	42.35	14.14	**20.02**	30.66	08.36	14.73
TextRank	32.87	13.90	20.93	18.67	**03.15**	12.23	26.55	08.64	19.01	36.93	13.60	**20.96**	34.50	10.86	17.42	24.41	08.32	13.44
LSA	29.23	10.47	18.35	18.70	02.60	11.82	25.12	07.74	17.26	33.55	09.00	16.02	32.65	09.22	16.36	22.68	08.09	11.73
PMI	36.56	15.49	23.11	19.13	02.89	12.45	**28.22**	08.51	**20.63**	37.82	10.85	18.33	-	-	-	-	-	-
SummPip	-	-	-	-	-	-	-	-	-	-	-	-	42.32	13.28	-	36.3	08.47	-
BERT SVD	25.28	7.60	15.90	17.09	02.44	11.41	22.14	05.60	14.77	33.85	09.43	16.45	40.86	13.42	18.44	18.57	03.76	10.27
SummVD	39.36	**17.70**	24.70	**19.7**	02.77	**12.70**	28.12	**09.27**	19.07	**38.06**	**14.49**	20.20	**43.55**	**15.83**	19.23	**37.80**	**10.15**	**16.43**

Table 3: ROUGE-1, ROUGE-2 and ROUGE-L F1 scores for every method described in Section 4.2 and SummVD described in Section 3.1 on every corpus described in Section 4.1. The best unsupervised method is bolded.

	NOUN	VERB	PROPN	NUM	ADJ	X	INTJ	PRON	ADP	SYM	PUN	DET	ADV
Source	21.3	12.6	5.5	1.7	6.7	0.2	0.2	6.7	10.9	0.1	7.1	8.1	4
After SVD	38	25.6	14	2.7	8.3	0.7	0.5	2.4	1.5	0	0.6	0	2.7

Table 4: Percentage of every POS tag in source documents vs top word on every axis after SVD.

jka, 2011) implementation of TextRank (Barrios et al., 2016) shows a huge gap in computation time when it comes to very large documents, SummVD being faster. Hence SummVD could be used for live summarization of large documents, daily news summarization, or even summarization of collection of documents.

6.3 SVD analysis

SVD is central to SummVD. Therefore it is crucial to understand how it affects the summarization process. In the analysis whose results are shown in Table 4, we count the POS tags of all the words in the source documents of every corpus used in our evaluation and the POS tags of every eigenvector top word, after the SVD has been applied. Looking at the differences in POS tags distribution between those two words sets can give a first idea of what kind of words the SVD tends to emphasize.

Table 4 shows that POS tags distribution in source documents differs widely from POS tags distribution in words selected after SVD. It shows that the SVD automatically selected most informative words : nouns, verbs, proper names and numbers and discarded less informative ones : adpositions, adverbs, interjections, without any frequency clue. In blue, the POS tags proportion emphased by SVD and in red the reduced ones.

6.4 BERT scores analysis

Using BERT as a sentence embedding method does not bring the best results as one can expect. Indeed, using the best BERT hidden layers configuration for text summarization achieve the results shown in Table 3. This difference compared to the GloVe based model can be explained by the fact that SVD is able to find the importance of a specific word in a document, while an interesting word can be dimmed in the general representation of the sentence embedding using BERT. This shows an interesting result : summaries might be based around the importance of specific words, which our method using SVD allows us to find.

7 Conclusion

This article presents a method, SummVD, based on word embedding and unsupervised methods which achieves fast and reliable summaries. We presented an extraction heuristic able to exploit the reduced document matrix that deals with single or multi-document and conducted an evaluation as complete as possible, led on heterogeneous corpora. The empirical study shows interesting results according to the state-of-the-art whether in terms of ROUGE effectiveness or in computation time. Compared to the most recent approaches, SummVD is better in average ROUGE scores while being around 1000 times faster on the datasets with the longest documents. This is achieved without any domain adaptation of the word embeddings; so there is room for improvement on domains such as medical/scientific or social media because they use a specific vocabulary that could be handled better. Its versatility on documents regardless of their type or size, paves the way to much more exploration on huge multi-document datasets, like Google, TripAdvisor or Amazon for example.

References

Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. Optics: Ordering

points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, page 49–60, New York, NY, USA. Association for Computing Machinery.

Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, page 550–557, USA. Association for Computational Linguistics.

Aurélien Bossard and Christophe Rodrigues. 2017. An evolutionary algorithm for automatic summarization. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 111–120, Varna, Bulgaria. INCOMA Ltd.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

H. P. Edmundson. 1969. New methods in automatic extracting. *J. ACM*, 16(2):264–285.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Mamdouh Farouk. 2018. Sentence semantic similarity based on word embedding and wordnet. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 33–37.

E. W. Forgy. 1965. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21:768–769.

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.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*, third edition. The Johns Hopkins University Press.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2341–2351, Melbourne, Australia. Association for Computational Linguistics.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 19–25, New York, NY, USA. Association for Computing Machinery.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

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

Ye Liu, Jian-Guo Zhang, Yao Wan, Congying Xia, Lifang He, and Philip S. Yu. 2021. HETFORMER: heterogeneous transformer with sparse attention for long-text extractive summarization. In *EMNLP (1)*, pages 146–154. Association for Computational Linguistics.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

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.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Jessica Ouyang, Serina Chang, and Kathy McKeown. 2017. Crowd-sourced iterative annotation for narrative summarization corpora. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 46–51, Valencia, Spain. Association for Computational Linguistics.

Paul Over and Walter Liggett. 2004. Introduction to DUC 2004: An intrinsic evaluation of generic news text summarization systems.

Vishakh Padmakumar and He He. 2021. Unsupervised extractive summarization using pointwise mutual information. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2505–2512, Online. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

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.

Josef Steinberger and Karel Jezek. 2004. Text summarization and singular value decomposition. In *Advances in Information Systems, Third International Conference, ADVIS 2004, Izmir, Turkey, October 20-22, 2004, Proceedings*, volume 3261 of *Lecture Notes in Computer Science*, pages 245–254. Springer.

Wenhao Wu, Wei Li, Xinyan Xiao, Jiachen Liu, Ziqiang Cao, Sujian Li, Hua Wu, and Haifeng Wang. 2021. BASS: boosting abstractive summarization with unified semantic graph. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6052–6067. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Jinming Zhao, Ming Liu, Longxiang Gao, Yuan Jin, Lan Du, He Zhao, He Zhang, and Gholamreza Haffari. 2020. Summpip: Unsupervised multi-document summarization with sentence graph compression. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1949–1952, New York, NY, USA. Association for Computing Machinery.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.