Comparing Semantic Models for Evaluating Automatic Document Summarization

Michal Campr(✉) and Karel Ježek

Department of Computer Science and Engineering, FAS,
University of West Bohemia, Univerzitn 8, 306 14 Pilsen, Czech Republic
{mcampr,jezek_ka}@kiv.zcu.cz

Abstract. The main focus of this paper is the examination of semantic modelling in the context of automatic document summarization and its evaluation. The main area of our research is extractive summarization, more specifically, contrastive opinion summarization. And as it is with all summarization tasks, the evaluation of their performance is a challenging problem on its own. Nowadays, the most commonly used evaluation technique is ROUGE (Recall-Oriented Understudy for Gisting Evaluation). It includes measures (such as the count of overlapping n-grams or word sequences) for automatically determining the quality of summaries by comparing them to ideal human-made summaries. However, these measures do not take into account the semantics of words and thus, for example, synonyms are not treated as equal. We explore this issue by experimenting with various language models, examining their performance in the task of computing document similarity. In particular, we chose four semantic models (LSA, LDA, Word2Vec and Doc2Vec) and one frequency-based model (TfIdf), for extracting document features. The experiments were then performed on our custom dataset and the results of each model are then compared to the similarity values assessed by human annotators. We also compare these values with the ROUGE scores and observe the correlations between them. The aim of our experiments is to find a model, which can best imitate a human estimate of document similarity.

Keywords: Contrastive opinion summarization · Summarization evaluation · ROUGE · Semantic models · TfIdf · LSA · LDA · Word2Vec · Doc2Vec · Document similarity

1 Introduction

In recent years, with rapid growth of information available online, the research area of automatic summarization has been attracting very much attention. Automatic document summarization aims to transform an input text into a condensed form, in order to present the most important information to the user. Summarization is a very challenging problem, because the algorithm needs to understand the text and this requires some form of semantic analysis and grouping of the

P. Král and V. Matoušek (Eds.): TSD 2015, LNAI 9302, pp. 252–260, 2015.
DOI: 10.1007/978-3-319-24033-6_29

content using world knowledge. Therefore, attempts at performing true abstraction (generating the summary from scratch) have not been very successful so far. Fortunately, an approximation called extraction exists and is more feasible for the vast majority of current summarization systems, which simply need to identify the most important passages of the text to produce an extract. The output text is often not coherent but the reader can still form an opinion of the original content.

A very challenging problem, which arises, is the evaluation of summarization quality. There are dozens of possible ways for the evaluation of summarization systems, and these methods can be classified basically into two categories [1]. *Extrinsic* techniques judge the summary quality on the basis of how helpful summaries are for a given task, such as classification or searching. On the other hand, *intrinsic* evaluation is directly based on analysis of the summary, which can involve a comparison with the source document, measuring how many main ideas from it are covered by the summary, or with an abstract written by a human.

Recently, one particular method has become very popular for the evaluation of automatic summarization. ROUGE [2] (Recall-Oriented Understudy for Gisting Evaluation) includes measures for automatically determining the quality of system summaries by comparing them to ideal human-made summaries. These measures count the number of overlapping units such as n-grams, word sequences, or word pairs between the system summary and the ideal summaries created by humans.

Since the evaluation of automatic summarization is based on a comparison between the system summary and a human-made one, we wondered if it is possible to utilize other NLP (Natural Language Processing) methods for evaluating the system summaries. In this paper, we examine some of the most popular NLP models and their performances in the task of assessing document similarity.

This paper firstly describes, what data we used for evaluating these models and how we annotated them. Then, we describe the models which we used in our experiments, each in its own section. Lastly, we provide the results and performances of chosen models in computing document similarities, and their comparison to human annotators.

2 Dataset

Our current research is focused on a specific variation of automatic summarization: contrastive opinion summarization. The main goal is to analyze the input documents, in our case restaurant reviews, and construct two summaries, one depicting the most important positive information and the other providing negative information. For this task, we constructed a collection from czech restaurant reviews downloaded from www.fajnsmekr.cz, in total of 6008 reviews for 1242 restaurants. For human annotation, however, this is too much, so we manually selected 50 restaurants, each with several reviews, so that their combined length is at least 1000 words. Three annotators then independently created two summaries for each restaurant, each with approximately 100 words.

This collection of gold summaries can already be used for evaluating our system summaries using the ROUGE metrics. However, we wondered, whether any other method could be utilized for this task, so we enhanced our collection in such a way, that it can be used for experimenting with document similarity. The main idea is to utilize the manually created summaries for finding the best algorithm for summarization evaluation. We will be investigating the similarity between those gold summaries the same way as if we were comparing the system summaries with the human-made ones. The process of additional annotation of our collection is described in the following subsection.

2.1 Data Annotation

We presented three annotators with 150 pairs of summaries, three positive summaries for each of the 50 restaurants. Each annotator was asked to assign a similarity score between them. The most obvious problem here is, how a human can come up with such a value after reading the texts. We devised a process of acquiring this score based on SCUs (Summary Content Units) used in the so called Pyramid evaluation [3] and combined it with a technique of annotation used in [4].

We asked our annotators to find pairs of facts which can be assigned the highest possible similarity score, according to our scale. Some can be pretty straightforward, such as those that praise the quality of service or food (assigned a value between 4 to 2), but other facts, that are missing or redundant in any of the summaries would be assigned with 0 or 1. These values are then averaged and thus the final score is assigned to the summary pair.

4 - Completely equivalent
3 - Mostly equivalent, differs in unimportant details
2 - Roughly equivalent, discussing the same topic, but important information differs
1 - Not equivalent, but roughly discussing a similar topic
0 - Different topics

3 Examined Language Models

In order to algorithmically perform a comparison of two documents, it is necessary to transform the original documents (plain text) into a representation, which a computer can understand, i.e a vector of features. The main problem is, what type of features to use. It is worth noting, that a common step for all models mentioned here is a preprocessing step, where the input string is tokenized into words and each word is lemmatized. The result is a set of terms corresponding to the input document. These terms can be used in several ways for constructing a model of the document. Among some of the more basic models are:

- Boolean model - equals to 1 if a term t occurs in document d and 0 otherwise
- Term frequency $tf(t, d)$ - raw number of times that term t is in the document
- Logarithmically scaled term frequency $\log(tf(t, d) + 1)$
- Augmented frequency - to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document

Besides those, there are more, however we decided to utilize only the most widely used and recognized models. In addition, we added to our experiments two relatively new ones, which are lately gaining much popularity (Word2Vec and Doc2Vec). Models used in our experiments are TfIdf, LSA, LDA, Word2Vec, Doc2Vec, and each one is briefly described in the following sections.

There is also the possibility of utilizing external linguistic resources, such as WordNet [5], for processing synonyms or other semantic information. However, our intention is to minimize the dependency of our methods on any language-specific tool. Also, we found a variety of colloquial expressions, which are not present in WordNet, and thus we decided not to use any such tool.

We should note beforehand, that (if not specified otherwise) the similarity score for each pair of documents is computed as the cosine similarity between two feature vectors v_1 and v_2:

$$sim(v_1, v_2) = \frac{\sum_{i=1}^{m} v_1[i] * v_2[i]}{\sqrt{\sum_{i=1}^{n} v_1[i]^2} * \sqrt{\sum_{i=1}^{n} v_2[i]^2}}, \tag{1}$$

3.1 TfIdf

The TfIdf (Term frequency - Inverse document frequency) weight of a term is a statistical measure used to evaluate how important a word is to a document in a collection or a corpus. Its importance increases proportionally to the number of times the term appears in the document, but is offset by its frequency in the corpus. The TfIdf weight of a term is a product of its frequency $tf(t, s)$ and inverse document frequency:

$$idf(t, D) = \log \frac{N}{|d \in D : t \in d|}, \tag{2}$$

where D is the document set, N is the size of set D and $|d \in D : t \in s|$ is the number of documents from D where the term t appears. The final value is then computed as:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D). \tag{3}$$

3.2 LSA

Latent Semantic Analysis (LSA) [6], also known as Latent Semantic Indexing (LSI), is a method for extracting and representing the contextual meaning of words by statistical computations performed on a corpus of documents. The underlying idea is that the totality of information about all the word contexts, in

which a given word does and does not appear, provides a set of mutual constraints that largely determines the similarity of meaning of words. The adequacy of LSA's reflection of human knowledge has been established in a variety of ways.

The creation of an LSA model starts with building a $m \times n$ matrix A, where n is the number of documents in the corpus and m is the total number of terms that appear in all documents. Each column of A represents a document d and each row represents term t.

There are several methods on how to compute the elements a_{td} of matrix A representing term frequencies, and among the most common are: Term frequency, TfIdf or Entropy. In our experiments, we present only models based on TfIdf, because they provided the best results.

With the matrix A built, LSA applies Singular Value Decomposition (SVD), which is defined as $A = U\Sigma V^T$, where $U = [u_{ij}]$ is an $m \times n$ matrix and its column vectors are called left singular vectors. Σ is a square diagonal $n \times n$ matrix and contains the singular values. $V^T = [v_{ij}]$ is an $n \times n$ matrix and its columns are called right singular vectors. This decomposition provides latent semantic structure of the input documents, which means, that it provides a decomposition of documents into n linearly independent vectors, which represent the main topics of the documents. If a specific combination of terms is often present within the document set, it is represented by one of the singular vectors.

3.3 LDA

Latent Dirichlet Allocation (LDA) [7] can be basically viewed as a model which breaks down the collection of documents into topics by representing the document as a mixture of topics with their probability distributions. The topics are represented as a mixture of words with a probability representing the importance of the word for each topic.

Since LDA provides probability distributions of topics, it is possible to use statistical measures for quantifying the similarity between two documents. There are many such measures, like KL-divergence, Hellinger distance or Wasserstein metric to name just a few. In our experiments, we chose to use the Hellinger distance (further denoted as LDA-h in the results) along with the cosine similarity to see, how those methods differ.

3.4 Word2Vec

Briefly, Word2vec is a two-layer neural net published by Google in 2013. It implements continuous bag-of-words and skip-gram architectures for computing vector representations of words, including their context. The skip-gram representation popularized by Mikolov [8], [9], [10] has proven to be more accurate than other models due to the more generalizable contexts generated. The output of Word2Vec is a vocabulary of words, which appear in the original document, along with their vector representations in an n-dimensional vector space. Related words and/or groups of words appear next to each other in this space.

Since Word2Vec provides vector representations only for words, we need to combine them in some way to get a representation of the whole document. This can be done by averaging all the word vectors for the given document, and thus creating just one document vector, which can be compared to another by cosine similarity (model designated as 'Word2Vec'). We also experimented with an n-gram analogy (denoted as 'W2V-pn'), i.e. combining word vectors for phrases with n words and then computing similarities between these phrase-vectors from both input documents.

In our experiments, we utilized the Word2Vec implementation (as well as Doc2Vec) in Python, called gensim, by Radim Řehůřek [11].

3.5 Doc2Vec

Googles Word2Vec project has created lots of interests in the text mining community. It provides high quality word vectors, however there is still no clear way to combine them into a high quality document vector. Doc2vec (Paragraph2Vec) modifies the Word2Vec model into unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. In [12], an algorithm called Paragraph Vector is used on the IMDB dataset to produce some of the most state-of-the-art results to date. In part, it performs better than other approaches, because vector averaging or clustering lose the order of words, whereas Paragraph Vectors preserve this information. Because of this, we also experimented with an n-gram analogy, i.e. computing Paragraph Vectors for phrases of length n and then comparing those phrases from both input document. The original model, which computes vectors for the whole documents is denoted as 'Doc2Vec' and the n-gram analogies are denoted as 'D2V-pn'.

4 Evaluation

As was described in section 2.1, we manually annotated 150 pairs of summaries with their similarity score, resulting in a total 450 human-made assumptions. Our main goal is to find such a model, that would provide the best correlation with human intuition. The annotated scores are based on a score scale with values ranging between 0 to 4, however in all the following texts and figures, they are converted into a 0-1 scale in order to be comparable with the models' scores. The average Pearson correlation coefficient between annotators is 0.8988.

Our results regarding the performance on document similarity assessment show, that the tested models can be basically divided into two groups:

1. models with no apparent correlation with human ratings
2. models showing significant correlation

The first group contains models: TfIdf, LSA, LDA and LDA-h. Figure 1 shows all these models in comparison to the averaged human-made values (dashed line). It is clear, that these models do not show any significant correlation, see Table 1

for exact values. This behaviour is most likely caused by the models' tendency to over-generalize the features, and by the fact, that they all work on the bag-of-words basis, effectively disregarding the word order. The Figure 1 shows, that these models compute very high values in all cases and do not provide scores from the full scale between 0 to 1, as are the human-made scores.

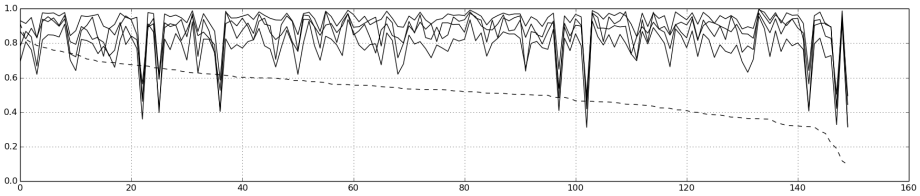

Fig. 1. Correlations of TfIdf, LSA, LDA and LDA-h models with average annotator scores (dashed line).

The second group of models contains: Word2Vec, Doc2Vec and their variations. These models show higher values of Pearson correlation with annotated data, see Table 1. Although Word2Vec does not show a very high correlation value (0.4009), it is apparent that it is able to capture a more sophisticated document structure. The same applies to Doc2Vec. Its base correlation (0.5523) with average annotated data shows, that its ability to take the word order into account provides a better document latent structure.

An interesting observation is, that methods comparing phrase-vectors (3.4) show a significant improvement over the base models. The best being W2V-p2 (0.4626) and D2V-p2 (0.6614).

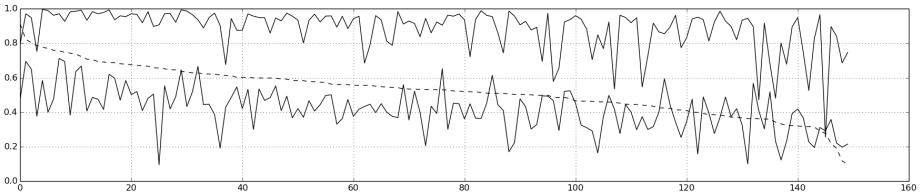

Fig. 2. Correlations of Word2Vec (higher values) and Doc2Vec models with average annotator scores (dashed line).

The last set of results was obtained using the ROUGE measures: ROUGE-1, ROUGE-2 and ROUGE-SU4. These measures are nowadays frequently used for summarization evaluation. From Figure 3 and Table 1 is apparent that these metrics provide the best overall correlations with annotated data, where the best one is 0.7276 for ROUGE-1.

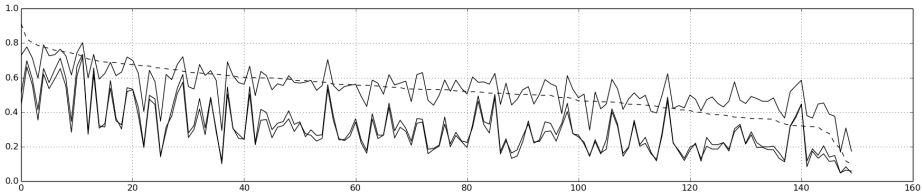

Fig. 3. Correlations of ROUGE metrics with average annotator scores.

Table 1. Pearson correlations between models and annotators. The 'avg' model score is computed against averaged scores from annotators.

	TfIdf	LSA	LDA	LDA-h	Word2Vec	Doc2Vec	ROUGE-1	ROUGE-2	ROUGE-SU4
a1	0.0430	0.0249	0.0890	0.0601	0.4111	0.4289	0.6376	0.5503	0.5237
a2	0.1152	0.0582	0.1651	0.1133	0.3777	0.5380	0.7248	0.6083	0.5814
a3	0.1287	0.0675	0.1678	0.1091	0.2742	0.5031	0.5633	0.5460	0.5271
avg	0.1103	0.0580	0.1613	0.1077	0.4009	0.5523	0.7276	0.6481	0.6209

	W2V-p1	W2V-p2	W2V-p3	W2V-p4	W2V-p5	D2V-p1	D2V-p2	D2V-p3	D2V-p4	D2V-p5
a1	0.3823	0.4042	0.3875	0.3912	0.3925	0.5491	0.5760	0.5330	0.5027	0.4759
a2	0.4623	0.4821	0.4707	0.4736	0.4785	0.6285	0.6585	0.6122	0.5724	0.5474
a3	0.3276	0.3430	0.3357	0.3361	0.3523	0.4368	0.5155	0.5195	0.4943	0.4859
avg	0.4410	0.4626	0.4493	0.4519	0.4420	0.6071	0.6614	0.6311	0.5954	0.5735

5 Conclusion

We explored performances of five language models (plus their variants) for computing document similarity. We aimed to find a model, which would best imitate the human estimates and, if successful, we could use this model for evaluation of automatic summarization along with the ROUGE measures. The best model proved to be the Doc2Vec (specifically D2V-p2) with score 0.6614. However, the best overall score was provided by the ROUGE-1 metric (0.7276), showing us, that there is still more research needed for semantic models to be able to outperform today's standard measures. Nevertheless, the Doc2Vec model shows promising results and we intend to conduct more experiments in this area.

Acknowledgments. This project was supported by grant SGS-2013-029 Advanced computing and information systems.

References

1. Steinberger, J., Ježek, K.: Evaluation measures for text summarization. Computing and Informatics **25**, 1001–1025 (2012)
2. Lin, C.: Rouge: A package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of the ACL 2004 Workshop, vol. (1), pp. 74–81 (2004)
3. Nenkova, A., Passonneau, R.: Evaluating content selection in summarization: The pyramid method. In: HLT-NAACL, pp. 145–152 (2004)

4. Eneko, A., Mona, D., Daniel, C., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Number 3, pp. 385–393 (2012)
5. Pala, K., Čapek, T., Zajíčková, B., Bartůšková, D., Kulková, K., Hoffmannová, P., Bejček, E., Straňák, P., Hajič, J.: Czech WordNet 1.9 PDT (2011)
6. Deerwester, S., Dumais, S., Landauer, T.: Indexing by latent semantic analysis. Journal of the American Society for Information Science **41**(6), 391–407 (1990)
7. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. The Journal of Machine Learning Research **3**, 993–1022 (2003)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop at ICLR, January 2013
9. Mikolov, T., View, M., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS (2013)
10. Mikolov, T., Yih, W.T., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of NAACL HLT (2013)
11. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, pp. 45–50. ELRA, May 2010. http://is.muni.cz/publication/884893/en
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of The 31st International Conference on Machine Learning, vol. 32, pp. 1188–1196 (2014)