Document Embedding for Scientific Articles: A validation of word embeddings

H.J. Meijer[1,2][0000−1111−2222−3333] and R. Karimi[2][1111−2222−3333−4444]

[1] University of Amsterdam, Science park 904, 1012WX Amsterdam, The Netherlands
[2] Elsevier, Radarweg 29, 1043 NX Amsterdam, The Neterlands
meijerarjan@live.nl, r.karimi@elsevier.com

Abstract. Over the last few years, word embeddings have taken a dominant position in the Information Retrieval domain. Many studies have been done concerning the quality and application of word embeddings on general texts, such as the Wikipedia corpus and comments on review websites. Giving promising results, the word embeddings have been studied and improved over recent years. However, these studies have been focused on generic texts, which are not limited to the characteristics of in-domain texts such as rare domain-specific words or have been focussed on small sets of academic texts. This research focusses on the quality and application of word embeddings on domain-specific texts, concerning a large corpus of 1.391.543 scientific articles which have been published in 2017. We validate the word embeddings using a categorization task to match articles to journals. We furthermore create a 2-dimensional visualization of the word embeddings on journal level, to visualize journal relatedness.

Keywords: Word Embedding · Document Embedding · Article Embedding · Journal Embedding · Embedding Visualization · Embedding Validation.

1 Data

For this research we used a dataset consisting of 186.962.354 tokens. These tokens have been collected from 1.391.543 articles which have been published in 3.759 distinct journals. In our dataset, each article is represented with a title and an abstract.

1.1 Sets

Embeddings For our research we used multiple embeddings (referred to as "sets"). These embedding sets are: the default embedding; as produced by the Word2Vec model and TF-IDF weighted embeddings (referred to in figures as *embedding*). We used 4 variants of TF-IDF weighted embeddings:

– TF-IDF embedding (*TFIDF_embedding*); all words, weighted with the TF-IDF score, averaged to create an article embedding.

- 10K embedding (*10K_embedding*); TF-IDF weighting limited to the top 10.000 most occurring tokens.
- 5K embedding (*5K_embedding*); TF-IDF weighting limited to the top 5.000 most occurring tokens.
- 1K 6K embedding (*1K_6LK_embedding*); TF-IDF weighting limited to the top 1.000 till 6.000 most occurring tokens.

TFIDF We furthermore used 3 TF-IDF sets, to create these sets we used the TF-IDF model and a hasher from the PySpark MlLib[3]. We controlled these sets in two ways, (a) adjusting vocabulary size of the input and (b) adjusting the number of hashbuckets. We label the TF-IDF sets as follows: "vocabulary size / number of hash buckets", we furthermore indicate 1.000 as 1K. Thus, we label the TF-IDF configuration that has a vocabulary size of 10.000 and 10.000 hash buckets as TF-IDF 10K/10K. To select the TF-IDF sets, we measured and memory usage of multiple TF-IDF configurations. These results can be seen in Figures 1 and 2, in this figure we can see that the performance on both title and abstract stagnates; the same is true for the memory usage, although the memory usage does not stagnate as fast as the performance. Given these results, we selected the 10K/10K, 10K/5K and 5K/5K configurations for our research.

2 Methodology

[1]

[3] `http://spark.apache.org/docs/2.0.0/api/python/pyspark.mllib.html`.

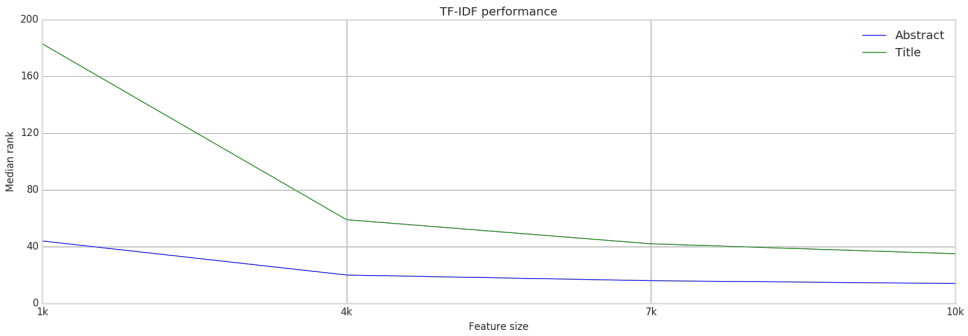

Fig. 1. TF-IDF performance on title and abstract.

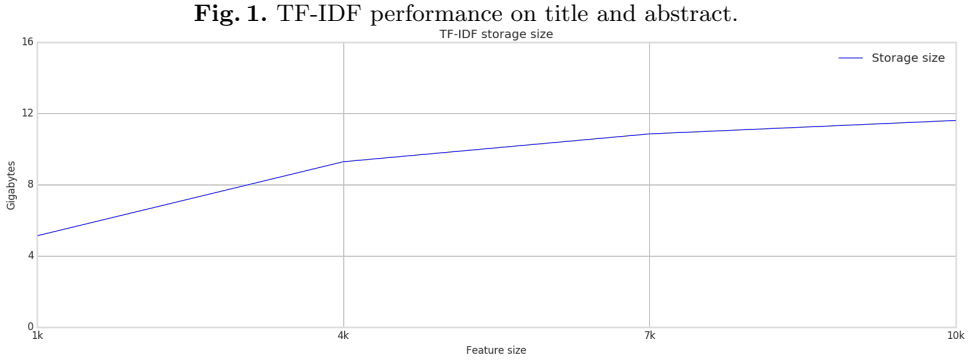

Fig. 2. TF-IDF memory usage for title and abstract combined.

Bibliography

[1] Bruni, E.: Men test collection (2012), https://staff.fnwi.uva.nl/e.bruni/MEN