Ranking Articles by Semantic Similarities
A Paragraph-Embedding-Based Approach

Liutong Zhou
lz2484
lz2484@columbia.edu
Columbia University
New York, NY

Meiqi Chi
mc4394
mc4394@columbia.edu
Columbia University
New York, NY

Leshen Sui
ls3452
ls3452@columbia.edu
Columbia University
New York, NY

Abstract

Paragraph embedding has been shown to achieve state-of-the-art results in many natural language processing applications. Motivated by Google's recent release of a document-embedding-based application, Talk to Books[1], we made an attempt to transfer the pretrained doc2vec model into another application, namely paper ranking. We experimentally showed that the paragraph-level semantic similarity measurement based on document embedding could achieve better results than counting-based methods. The experiment achieved promising results in terms of how well the machine's ranking aligns with humans' judgment. This approach can greatly facilitate the process of literature review by recommending the most related papers to researchers, taking not only key words but also the semantic meanings of articles into account.

Key words:

document embedding, topic modeling, paper ranking

1. Introduction

In academia, it is not uncommon that a researcher comes up with an idea based on a paper, searches for potentially topic-related papers in an online database, spends a significant amount of time on the search-filter-search iterations, but still gets overwhelmed by the necessity to manually go through a large number of papers to decide on important references. This is a challenging task for a novice researcher who just entered a new research field, and is a tedious task even for experienced researchers. Though a literature review ought to be extensive, it will make the researcher's life easier if the machine can filter and rank the candidate papers by how much related they are to the core idea, so that researchers can focus more on the most related content instead of being distracted by unrelated research articles.

The need of filtering out unrelated articles and rank related articles in order of importance could possibly be met by measuring the semantic similarities of paragraphs. Motivated by recent researches in document embedding [2] and Google's successful release of Talk to Books[3], we made an attempt to incorporate these state-of-art paragraph vectorization methods into the application of paper ranking.

The goal of this project is to apply the most recent document embedding methods to vectorizing paragraphs and measure the semantic similarities of articles. This project has significantly practical implications, one of the application scenarios is automatically filtering out unrelated research articles and rank related articles in order of importance. Different from the pervasive methods for measuring lexical similarity (eg. measuring text similarity by longest common substrings or number of common n-grams), this project focuses on measuring semantic similarity at paragraph level.

This paper is structured as follow. In section 2, we first review the development of vectorization methods in the natural language processing (NLP) research community. In section 3, we introduce doc2vec, a neural-network-based method to embed paragraphs into fixed-length vectors, on top of which many NLP tasks, including sentiment analysis and topic modeling, have been shown to achieve state-of-the-art results. In Section 4, we test different vectorization methods in a topic modeling task, and compare their performances. Then we present the promising results that we achieved in paper ranking based on document embedding. Finally, we discuss the limitations of the work and point out the directions for future improvement in section 5.

2. Literature Review

Traditionally, counting-based word vectorization methods, also known as Bag of Words (BOW), were widely used for text analysis, which usually introduce high dimensionality and sparsity, but still provides limited capacity to capture semantics. For example, a n-gram vectorization of a document usually results in a large sparse matrix. As apposed to the traditional distributional methods, the advent of distributed models [4] provides a condensed representation of words with more capacity to capture semantic meanings of texts, thus achieve better generalization. This technique is known as word embedding, and has been largely popularized by the implementation of two algorithms, CBOW and skip-grams in the package word2vec [5], since 2013.

Whereas a n-gram model works in terms of discrete units that have no inherent relationship to one another, a continuous space model works in terms of word vectors where similar vectors indicate similar semantic meanings.[6] To obtain embeddings for longer pieces of texts such as sentences, averaging the embeddings of words in a sentence has proven to be a surprisingly successful and efficient method [7]. By directly including the averaged embeddings of the words in the cost function, the problem that word embeddings alone are not optimized for the task of sentence representation is solved. The restriction here is that document needs to be split into sentences and the order of sentences needs to be preserved.

In analogy of word embedding, paragraph embedding is proposed.[8] It is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Similar to word2vec, there are also two kinds of paragraph embedding algorithms namely Distributed Bag of Words of Paragraph Vectors (PV-DBOW) and Distributed Memory of Paragraph Vectors (PV-DM)[2]. The former uses only one paragraph vector to represent the text without word ordering while the latter combines both. Both algorithms are implemented efficiently in the Gensim package[9]. We directly use the implementations in Gensim for this project.

In section 3, we introduce the details of the PV-DM algorithm that we used in this project. Then we experimentally show that the PV-DM algorithm works best for capturing the semantic meanings of texts.

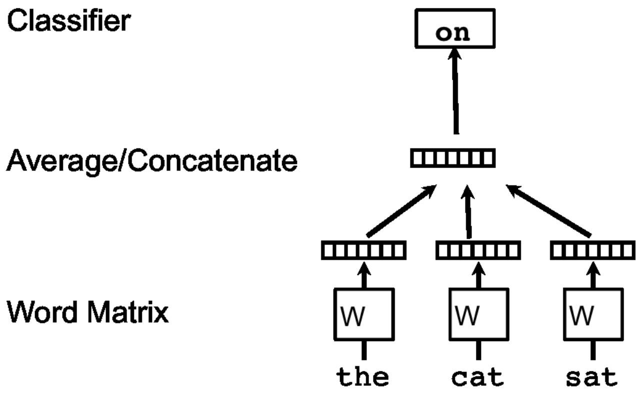
3. Methods

This section introduces the PV-DM algorithm for document embedding in detail, and the approaches we used for measuring and ranking documents

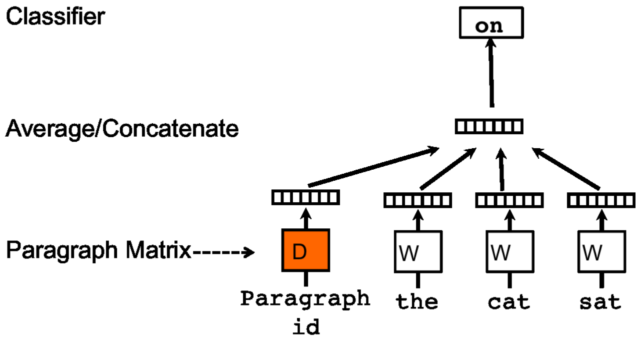
3.1. Document Embedding

The way of learning a word embedding matrix W is illustrated in figure 1 (a). If the vocabulary size is n and each word is to be embedded into a d dimensional vector space, then W is a n $\times$ d matrix ($W \in \mathbb{R}^{n \times d}$). Multiplying the one-hot encoded word vector (assuming a row vector) by W on the right embeds the word into a d dimensional vector. To find the coefficients of W, form a classifier which takes a context of words as input and outputs a target word. For example, the context can be defined as "the previous 3 words", and the target can be "the next word". In this setting, the first (input,output) pair is extracted to be (['the','cat','sat'],['on']) when given a corpus "the cat sat on a tree.". The second (input,output) pair would be extracted as (['cat','sat','on'],['a']). A multilayer perceptron (MLP) with an input layer (also called embedding layer), a hidden average layer (or concatenation layer) and a softmax output layer fulfills the requirement as a classifier. W is obtained by training this classifier on a large corpus using stochastic gradient descent (SGD).

Similar to training a word embedding matrix W, the paragraph vector can be obtained in the same way by training the same classifier under the same setting of a classification problem. The only modification to the architecture is an addition of paragraph id, which is mapped to a vector by the paragraph embedding matrix D. The paragraph can act as a memory of the topic of the paragraph and provides missing information from the current context window. At training time, all the coefficients including D and W are jointly updated using SGD. At prediction time, an unseen paragraph needs to be encoded. This is achieved by appending a new row to D, fixing all the coefficients of the trained model and only update the newly added row in the same classification problem.



(a) Learning word vectors



(b) Learning paragraph vectors

Figure 1: The PV-DM algorithm for document embedding. (a) illustrates the architecture of a MLP classifier for learning word vectors. (b) adds an additional embedding matrix D to encode paragraph ids into vectors. If there are m paragraphs in the corpus, D would have a shape of m by p where p is the dimension of paragraph vectors. Figures are cited from [2]

This algorithm is referred to as the PV-DM algorithm and is used in this project for embedding documents into vectors. Hereafter, we refer to this method as doc2vec in analogy to the well-known word2vec method.

3.2. Evaluation Methods

To better understand how well the doc2vec method works in practice in capturing semantics of documents, we incorporate BOW and word2vec as baselines. We will benchmark different vectorization methods quantitatively in a topic modeling task, and qualitatively by visualization and eye screening.

Topic modeling is a conventional problem in NLP and information retrieval research. It is an unsupervised learning problem that input a corpus of documents, output a group of underlying topics along with the probabilistic distributions of each document being classified in each topic. Specifically, we apply the Latent Dirichlet Allocation (LDA) algorithm [10] on the vectorized documents to conduct topic modeling. We expect that a good document vectorization algorithm should perform well in the task.

We use two metrics to evaluate the topic modeling results, which are normalized mutual information (NMI) and adjusted rand index (ARI). We briefly describe their definitions. Let U denote the predicted labels N objects and V denote the ground truth labels. $P(i) = |U_i|/N$ calculates the empirical probability of a randomly picked object falling into category U_i. Similarly for $P'(j) = |V_j|/N$. $P(i,j) = |Ui \cap Vj|/N$ calculates the probability of a randomly picked object falling into both category U_i and category V_j, meaning $U_i = V_j$. The mutual information (MI) between U and V is defined as follow:

$$\begin{aligned} MI(U,V) &= \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\; ln\left(\frac{P(i,j)}{P(i)P'(j)}\right) \\ &= \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\; ln\left(\frac{N|U_i \cap U_j|}{|U_i|\;|U_j|}\right) \end{aligned} \tag{1}$$

Normalizing MI by the entropies of U and V, we get the NMI:

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}} \tag{2}$$

where $H(U)$ and $H(V)$ are the entropy of U and V respectively:

$$H(U) = -\sum_{|U|}^{i=1} \frac{|U_i|}{N}\; ln\left(\frac{|U_i|}{N}\right) \tag{3}$$

$$H(V) = -\sum_{|V|}^{j=1} \frac{|V_j|}{N}\; ln\left(\frac{|V_j|}{N}\right) \tag{4}$$

Let a denote the count of pairs of objects assigned in the same category in U and also assigned in the same category in V. Let b denote the count of pairs of elements not assigned in the same category in U and not assigned in the same category in V either. Rand index (RI) is defined as follow:

$$RI = \frac{a+b}{\binom{N}{2}} \tag{5}$$

adjusting the RI by chance, we get ARI:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{6}$$

The closer NMI is to 1, the better the prediction aligns with ground truth. The same statement holds for ARI.

3.3. Ranking Articles by Semantic Similarity

The success of word2vec in capturing semantic meanings of words is evidenced by the example:

$$\text{King} - \text{Man} + \text{Woman} = \text{Queen}$$

This analogy task is achieved by embedding words on the left side of the above equation into vectors and do the following calculation:

$$\text{result} = arg \min_{x \in vocab} dist\left(vec(\text{'King'}) - vec(\text{'Man'}) + vec(\text{'Woman'}),\ vec(x)\right)$$

where $dist$ is usually the cosine distance:

$$\text{cosine\_distance}(\vec{x}, \vec{y}) = 1 - \frac{< \vec{x}, \vec{y} >}{\|\vec{x}\| \ \| \vec{y}\|} \tag{7}$$

We follow the convention and use cosine similarity to measure document similarity.

$$sim(\vec{x}, \vec{y}) = \frac{< \vec{x}, \vec{y} >}{\|\vec{x}\| \ \| \vec{y}\|} \tag{8}$$

where $\vec{x}$, $\vec{y}$ are the two document vectors.

We calculate the similarity scores pair-wisely for each pair of documents in the corpus to produce a similarity matrix S and query the top k most similar articles from the matrix.

4. Experiments

4.1. Description of Data

Our data was collected from Cornell University Library through arxiv.org API. This API is used for querying research paper using topic keywords. For this project, we carefully chose five topics: philosophy, aerospace, electron, medical and machine learning. For every topic, we queried the most related 100 papers and extracted their topics and abstracts for further analysis. The reason we selected the above topics lies in two aspects: on one hand, these are hot areas for researches so we can obtain enough relevant data; on the other hand, we believe there are gaps between these research areas, therefore, we expected the result of our model would reflect such differences.

4.2. Visualizing Document Vectorization Results

Using t-SNE [11] and PCA, we are able to reduce the high dimension document vectors produced by three different methods to 2 dimensions and visualize them in Figure 2. Compared with PCA, t-SNE produced better visualization results with doc2vec and Averaging Word Vectors as papers in the same topic are grouped more closely together. Among the three document vectorization methods, doc2vec seem to capture the semantic meaning accurately. The cluster of papers in philosophy is the most concentrated and has a much larger margin than others, which is reasonable since it is the most distinct topic among all. The other four topics are related to some degree, so there is some overlapping.

4.3. Evaluating Document Vectorization Methods by Topic Modeling

The results of topic modeling based on different document vectorization methods are summarized in Table 1. We use NMI and RAI to directly measure how well the predicted topics align with the ground truth, indirectly reflect how well the document semantic meanings are captured by each vectorization method. It can be seen that doc2vec greatly outperforms simple average of word vectors and BOW method Tf-idf. Therefore, we only use doc2vec to compute similarity scores and make recommendations.

Table 1: Topic Modeling Validation Results

Vectorization Method	NMI	RAI
Doc2vec	**0.694**	**0.613**
Tfidf	0.014	0.004
Word2vec with averaging	0.202	0.061

4.4. Paper Ranking by Semantic Similarities

We calculated the similarity matrix S, and then provided a list of recommended top 5 papers. A sample result is listed in table 2. After qualitatively analyzing the results, we can see that the recommendation is quite reliable.

Table 2: Paper ranking results: Finding topic-similar articles by document semantic meanings

Topic	Paper Title	Similarity
aerospace	Relativistic description of the double P-wave charmonium production in e^+e^- annihilation	-
aerospace	Relativistic corrections to the pair double heavy diquark production in e^+e^- annihilation	98%
aerospace	Relativistic corrections to double charmonium production in high energy proton-proton interaction	98%
aerospace	Relativistic corrections to $_c$-pair production in high energy proton-proton collisions	98%
aerospace	Pair double heavy diquark production in high energy proton-proton collisions	98%
philosophy	The Interpretation of Wave Mechanics with the help of Waves with Singular Regions	-
philosophy	Elementary Considerations on the Interpretation of the Foundations of Quantum Mechanics	99%
machine learning	Identifying and Harnessing the Building Blocks of Machine Learning	-
machine learning	Probabilistic Matrix Factorization for Automated Machine Learning	94%
electron	Electron pairing: from metastable electron pair to bipolaron	-
electro	Metastable electron-pair states in a two-dimensional crystal	94%
electron	Effects of Electron-Electron Scattering on Electron-Beam Propagation in a Two-Dimensional Electron-Gas	-
electron	Pair emission from bare magnetized strange stars	94%

5. Discussion

5.1. Conclusion

We presented three methods, Bag of Words, Averaging Word Vectors and doc2vec to encode documents and capture their semantic meanings. Our experiment results show that doc2vec gives the best performance in computing the similarities between documents and clustering documents into different topics. This is consistent with our expectation as doc2vec preserves the order of words, shares the power of word2vec and also operates on the paragraph level. By measuring the similarities between document vectors of research papers, we are able to filter out a list of papers that are most related to the objective paper. This can greatly facilitate the process of literature review.

5.2. Future Work

Although the focus of this work is on research papers, this method can be generalized to find out related articles in many other different genres. In other categories where precise abstract is not available, the whole document can be used to train the document vector. As long as the objective document has a clear topic, we expect the ranking articles produced will be closely related to it in semantic meaning.

6. Executive Summary

6.1. Milestones

1. Query research articles of various topics from Cornell University Library via arxiv.org API.
2. Data cleaning and pre-processing: Parsing the XML format data returned from the API.
3. Use Doc2vec to obtain the Document Vector for each paper, and then calculate similarity scores of each pair of papers, rank related articles based on descending similarity scores.
4. Apply tf-idf, word2vec to derive the vector of each paper, and repeat the same procedure as above.
5. Visualize the vector of each document using different methods, including tSNE & PCA.
6. Adopt LDA to the document vectors derived by all three methods and compute normalized mutual information score and adjusted rand index score for comparison and evaluation.

6.2. Obstacles

Training paragraph vectors are computationally expensive, even when aided by transfer learning. Thus, we run Doc2vec model on AWS, making use of GPU parallelism to improve efficiency.

6.3. Define Success

1. Offline evaluation: the Doc2vec model achieve an NMI of 0.694 and RAI of 0.613, which is much better than Tfidf and Word2vec model.
2. Online evaluation: The articles with high similarity should refer to similar topics, based on human judgment.

7. Acknowledgment

This project is based on tools and techniques introduced in the course *COMS W 4121 Computing Systems for Data Science*, including using NoSQL databases for storing and managing documents, cleaning and preprocessing dataset running deep learning models on an AWS cloud virtual machine.

References

[1] Research Blog: Introducing Semantic Experiences with Talk to Books and Semantris.

[2] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal Sentence Encoder. 3 2018.

[4] Yoshua Bengio, Holger Schwenk, Jean-Sbastien Senécal, Frderic Morin, and Jean-Luc Gauvain. Neural Probabilistic Language Models. In *Innovations in Machine Learning*, pages 137–186. Springer-Verlag, Berlin/Heidelberg, 2006.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. 1 2013.

[6] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

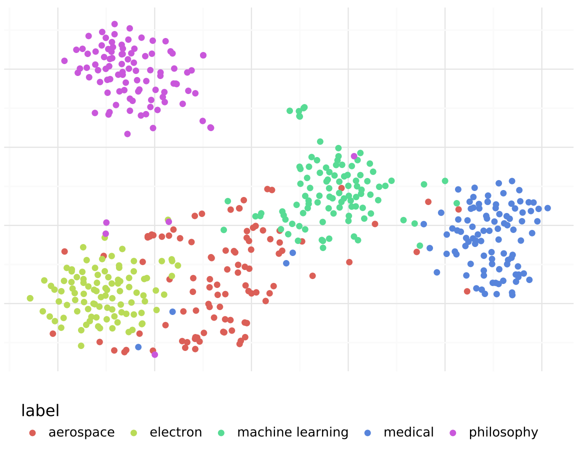
[7] Tom Kenter, Alexey Borisov, and Maarten de Rijke. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. 6 2016.

[8] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*, 2015.

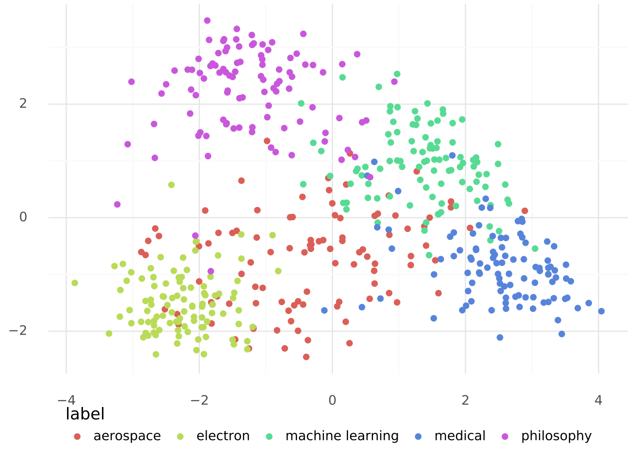
[9] Radim Reh Urek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 5 2010. ELRA.

[10] David M Blei, Blei@cs Berkeley Edu, Andrew Y Ng, Ang@cs Stanford Edu, Michael I Jordan, and Jordan@cs Berkeley Edu. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

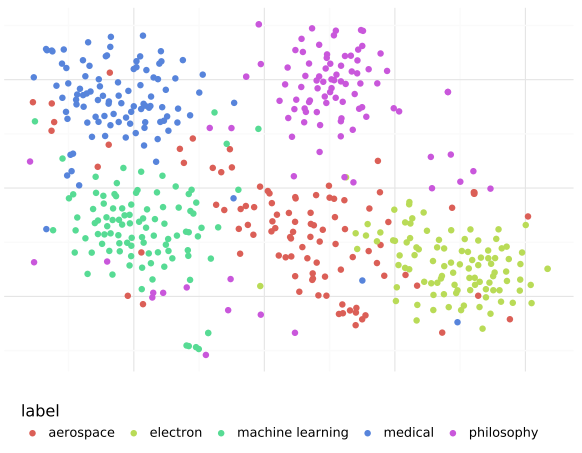
[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.



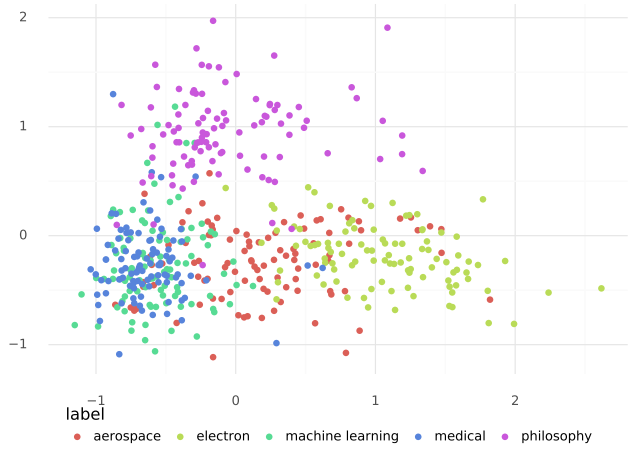
(a) Doc2vec + t-SNE



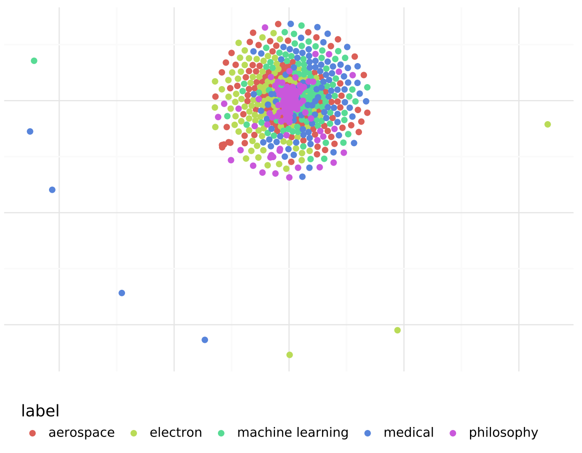
(b) Doc2vec+PCA



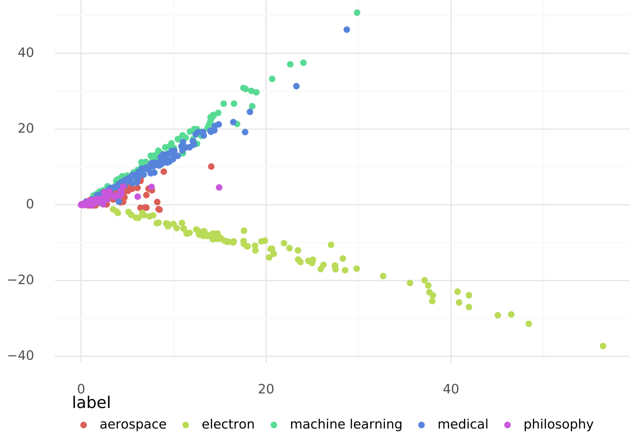
(c) Word2vec+Averaging+t-SNE



(d) Word2vec+Averaging+PCA



(e) Tfidf + t-SNE



(f) Tfidf + PCA

Figure 2: Documents visualized in 2-dimensional space using different vectorization methods. t-SNE (left column) and PCA (right column) are used for projection. Document vectors are colored by their topics.