Vectorization of Documents and Analysis of their Identity Using a Neural Network

Anton Rogozin[1], Marina Medvedeva[2] and Vitaly Ford[3]

[1] Ural Federal University, Ave. Lenina, 51, Yekaterinburg, Sverdlovsk region, 620075, Russian Federation
rogozin.anton96@gmail.com
[2] Ural Federal University, Ave. Lenina, 51, Yekaterinburg, Sverdlovsk region, 620075, Russian Federation
marmed55@yandex.ru
[3] Arcadia University, 450 S Easton Rd, Glenside, PA 19038, 620075, USA
fordv@arcadia.edu

Abstract. The purpose of this article is to design a convenient and fast system for searching of similar documents. The natural language processing is dedicated to this. The system is based on the idea of vectorization of documents. The optimal strategy for selecting doc2vec hyperparameters based on Word2vec for the best quality of searching of similar documents is described and tested in practice. Word2vec collects statistics on the joint appearance of words in expressions and solves the problem of reducing dimension. Word2vec is based on a two-layer neural network. The model produces compact vector representations of words that reflect the relations of these words in processed texts. This approach may give an accuracy of about 90%. Vectorization of documents has appeared recently, but this experience provides great potential for research and practical application in the classification and search of texts.

Keywords: Word Embedding, NLP, Natural Language Processing, doc2vec, Word2vec, LSA, PLSA, LDA, t-SNE, Data Science, Big Data, Neural Network, Higher Education, Experience.

1 Introduction

Textual data is widespread. The Internet is a huge collection of texts in various formats. These texts allow people to solve a number of problems. These tasks include predicting the rating based on the text content of a post, determining the emotional color of a comment, classifying news by history, generating text, as well as searching for similar news.

These tasks were difficult to solve with the help of computing devices, since there was not enough knowledge for training and technologies that would allow processing large data in the acceptable amount of time previously.

The rapid growth in the volume of digital data began in the 20s [1]. This growth made it possible to obtain a huge array of textual data, starting from converting books

to text files, gradually converting the format of printed newspapers to the format of online newspapers, to text discussions on forums and social networks. There has been a rapid increase in computing power that would allow us to process such a volume of data in the reasonable amount of time. Thanks to such a volume of data and the development of technology, it has become possible to analyze such volumes of data quite quickly, for example [2, 3].

The task is to learn how to use text data in machine learning algorithms. The main problem is that text is a collection of variable-length objects made up of characters.

The words "go" and "move" are synonyms, and these have the similar meaning. These words are made up of different characters. The computer cannot determine whether these words have the same meaning by their characters.

When analyzing texts, it is necessary to evaluate how similar the various pairs of words are. This can be done using text data. It turns out that words with similar meanings are often found next to the same words. In other words, similar words have the same contexts. The Word2vec method is based on this principle.

Google introduced the Word2vec algorithm in 2013, which gradually began to gain popularity. The theory behind it has shown its effectiveness. Word2vec takes into account the context of the word and at the same time reduces the amount of data [4].

Word2vec collects statistics on the joint appearance of words in expressions and solves the problem of reducing dimension. Word2vec is based on a two-layer neural network. The model has at its output compact vector representations of words that reflect the relations of these words in processed texts.

The probability of the word closeness with another word ω_j is shown in the equation (1):

$$\rho(\omega_i \mid \omega_j) = \frac{\exp(\langle \overrightarrow{\omega_i} \mid \overrightarrow{\omega_j} \rangle)}{\sum \exp(\langle \overrightarrow{\omega_i} \mid \vec{\omega} \rangle)} \tag{1}$$

This function shows how likely it is to meet the word $\overline{\omega_t}$ in the context of the word $\overline{\omega_j}$, i.e. nearby. Vector representations of words is adjusted so that the probabilities of meeting words in the same context are as high as in equation (2). The quality functional for each word includes the probabilities of meeting it with the words before and after it:

$$\sum_{i=1}^{n} \sum_{j=-k}^{k} \log(\rho(\omega_{i+1} \mid \omega_j)) \to \max \tag{2}$$

This functionality may be optimized using stochastic gradient descent.

Vectorization of the text is obtained by averaging the product of the vectors of all words multiplied by their weights included into the document [5, 6].

Using the averaging of vectors described above, word order is ignored. To work with texts of variable length, doc2vec was developed [7]. This method is similar to Word2vec. This model is more general due to the added paragraph and document vectors.

Doc2vec is a very good approach. It is easy to use and gives good results. According to the studies of the quality of different models, Word2vec shows better search results for similar documents on a larger scale than LSA, PLSA, LDA [8].

For example, this system can be useful in classifying letters for the technical support service of the university. The system shows similar letters and the answers to them, so that the specialist spends less time on decision.

2 Purpose

The purpose of this article is to design a convenient and fast system for finding similar texts.

3 Tasks

It is worth doing the following tasks:

- Text processing.
- Studying the theory of word embedding.
- Designing an algorithm to determine the optimal model parameters.
- Model development.
- Application of the model in practice.
- Discussion of the results of the designed model.

4 Text Processing

Texts must be prepared before training the model. Text processing consist of:

- Text pre-clearing.
- Tokenization.
- Normalization.
- Delete stop words.

Text pre-clearing consist of:

- Removing formatting and line breaks.
- Typos correction.
- Removing non-textual information.
- Merge short texts.

Tokenization is the breaking of a continuous line into separate words.

Tokenization consists of several stages. The first step is to lowercase the text. The next step is to replace all punctuation and other characters with spaces. The last step is that each word, separated by a space, is declared a separate token.

Words that are too common do not help to determine the subject of the text and are called stop words. Too rare words also do not help. Words that are less common about

ten times are simply removed from the collection. Removing stop words and words that are too rare can reduce the vocabulary and reduce the computational complexity of the task.

The next stage is the normalization of words in the text, in other words, the reduction of each word to its base/initial form. There are two main approaches to normalization: stemming and lemmatization.

Stemming is the easiest normalization method. This approach consists of cutting off the end of each word according to certain rules.

Lemmatization uses a dictionary in which a large number of words and their forms are already recorded. The first step is to check the word in the dictionary. If the word is not in the dictionary, lemmatization according to a certain algorithm normalizes the word.

It has been found that lemmatization works much longer (about 80 times) than stemming. This is due to the fact that lemmatization uses the more complex algorithm for obtaining the initial form, when stemming simply cuts off the endings of words according to the limited set of patterns.

Therefore, stemming is used in this article.

5 Learning

A typical task of supervised learning is to create a black box (model), which receives some information about the object on the basis of which the forecast is based. This forecast that we may use in practice.

The black box is some model. In order to work efficiently, the model needs to be set up and train with the necessary parameters on the training set. This can be achieved by training the model several times with different parameters and comparing the error of these models. After these actions, the best model may give a good forecast for an unfamiliar object with a fixed set of features.

Overfitting is a phenomenon when the constructed model explains well the examples from the training set, but does not work well on the examples that did not participate in the training.

Underfitting is a situation when the algorithm poorly describes both the training set and new data. In this case the algorithm needs to be complicated.

Since doc2vec is a neural network, it uses a standard approach for this kind of model, which avoids overfitting.

It is necessary to use a test data set. The main data set is divided into two parts. The first data set is used to train the algorithm, and the second data set is a test dataset. Typically, the main data set is divided in the ratio of 80/20.

The advantage of the test data set is that the algorithm may be trained only once, but the result depends greatly on how the partition was performed.

Hyperparameters are those parameters of the algorithms that cannot be obtained from the training set during training; therefore, these ones should be selected by training repeatedly the algorithm.

Doc2vec is a ready-made implementation of a two-layer neural network. Doc2vec hyperparameters are the length of the document vector and the window within which the model searches for the context of the word.

The quality metric is understood as how accurate the model provides answers to test data. The quality metric is the antithesis of the error. Quality metrics are maximized, and the error is minimized.

It is necessary in the search results of similar documents as much as possible.

It is possible to use the arithmetic mean precision, which is calculated as the quotient of the sum of all objects on the accuracy of the test sample by the number of objects in a test sample, as in the equation (3):

$$AP@k(q) = \frac{\sum_{i=1}^{k} Precision@k(q)}{k} \tag{3}$$

Precision calculates the accuracy for the first k query documents q as in the equation (4):

$$Precision@k(q) = \frac{1}{k}\sum_{i=1}^{k} y(q, d_q^i) \tag{4}$$

$y(q, d_q^i)$ returns one if d_q^i matches query q, otherwise zero is returned. By query is meant the issuance of k similar documents calculated by the cosine metric for a document.

The error is equal as in the equation (5):

$$1 - AP@k(q) \tag{5}$$

Usually, optimization methods are used to minimize model errors, which allow one to select hyperparameters so as not to train the model repeatedly at all possible parameter values. This approach is based on iterations, repeated training of the model, in which the error decreases with each training. This approach can significantly speed up the selection of optimal hyperparameters. Most of these methods work with floating point increments rather than integer increments. Doc2Vec accepts only integer values.

The problem arises of enumerating all possible values of hyperparamters due to these features. To avoid learning at all possible values, a certain range of values is selected. This range is selected for practical reasons. It is used for grid search.

Iterating over the entire list of vectors is extremely slow. There is Approximate k-Nearest Neighbors (AKNN) to solve this problem. This algorithm is faster than honest search. AKNN returns close neighbors, but not with 100% certainty of the closest.

There were 32-dimensional vectors, the training data consists of 800,000 vectors; the test data set consists of 100,000 vectors for performance testing.

The Nearest Neighbors algorithm from sklearn worked in 3 minutes 13 seconds for search in all vectors. This search time is equal to the sum of searches for similar documents for each document. The hnsw algorithms from the nmslib library and ivf from the faiss library give an accuracy of 99.8 - 99.9% and 30-34 seconds. The AKNN algorithm consists of building indexes, which takes about 28-30 seconds, and searching by vectors, which works for about 2-4 seconds.

AKNN algorithms work compared to the honest algorithm KNN, which searches for similar vectors across all documents, six times faster. If we assume that the indices in the AKNN algorithms have already been built, then the AKNN algorithm is 100 times faster than KNN.

These models will be visualized. The task of data visualization is a special case of nonlinear dimensionality reduction when data is projected onto a plane. The data is projected so as to preserve the entire structure and patterns. A good solution is the t-SNE method.

Model training and writing code in Python is carried out using the Google Collaboratory environment.

Colaboratory is Google's cloud platform for advancing machine learning technology. The system provides a virtual machine for free with installed popular libraries TensorFlow, Keras, sklearn, pandas, etc. Colaboratory lets people work with notebooks like the Jupyter. Notebooks are stored on Google Drive. The notebook is created for 12 hours in the Docker container, which is Linux with Google Collaboratory installed on it.

Standard libraries for working with data, training, visualization are used:

- numpy is a library for convenient work with arrays.
- pandas is library for working with data frames.
- sklearn is library containing various classification, regression, clustering, downsizing algorithms using t-SNE.
- nltk is a package of libraries and programs for symbolic and statistical processing of a natural language.
- pymystem3 is a library for lemmatizing tokens of the Russian language.
- scipy is a library for performing scientific and engineering calculations.
- matplotlib is a library for multidimensional data.
- genism is a library containing the implementation of doc2vec.

6 Data sets

A toy dataset is a small dataset. This set consists of 70 documents. Each document consists of 3-5 words. The set consists of 3 categories. The text corpus consists of 300 Russian words. The texts are about wood, construction and the computer. Each category has its own unique words that do not overlap with other categories. At least one word is repeated in each text of a separate category.

The data set with Russian books consists of 700 documents of various genres. Each document consists of with an average of 100 thousand words. The set consists of 8 genres. Genres are fantasy, fiction, mysticism, detective story, pulp fiction, cooking,

classic, non-fiction. Each document represents a book with a lot of words. The document may be a novel or a collection of short stories. The body of the text consists of 75 million words.

The set of Russian news from the Lenta.ru site consists of 600 thousand documents. Each document has 200-300 words. The text corpus consists of 120 million words. Each news is written with an introduction to the course of affairs. The dataset contains news from 2000 to 2018.

7 Results

This experiment should have been unsuccessful. The doc2vec model requires more than 5-10 million words for normal learning [9].

All documents have passed the text processing stage. Then the model is repeatedly trained. The following graph of the dependence of the error on the length of the vector is obtained (see Fig. 1).

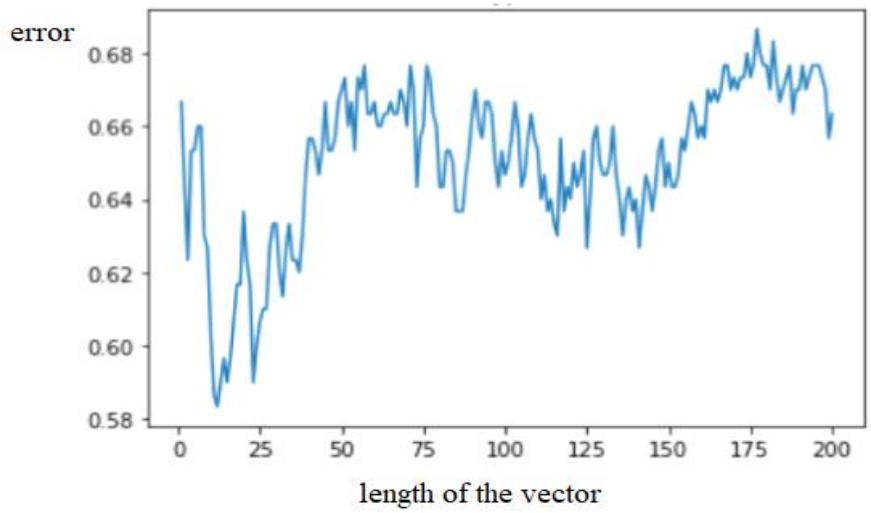

Fig. 1. Dependence of the error on the length of the vector.

The pattern on this graph is poorly traced due to the scatter of values. The error decreases sharply, reaches its global minimum, after which it slowly grows or remains at the same level.

The minimum error is achieved with a vector length of 16. The model with a minimum error cannot find similar documents in such a small data set. It is impossible to unequivocally divide the resulting large cluster with a uniform distribution of objects into three document clusters (see Fig. 2).

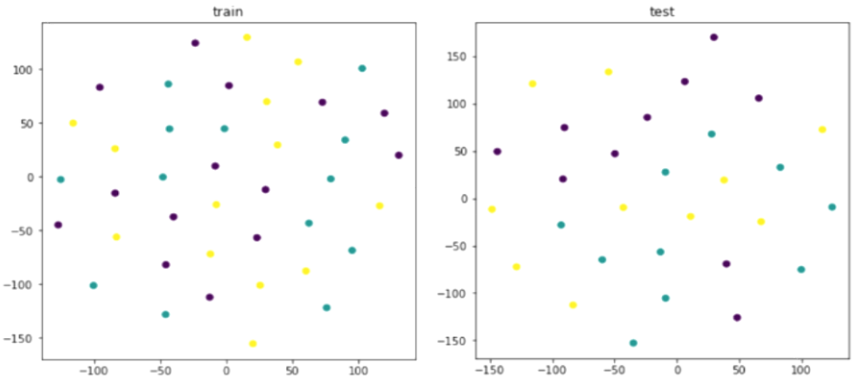

Fig. 2. Visualization graphs on the train and test data sets.

Much more data is needed to determine the relationships between words and to convert tokens into a vector of documents.

700 books of various topics are selected. Topics include fantasy, fiction, mysticism, detective story, pulp fiction, cooking, classic, non-fiction.

All documents have passed the text processing stage. Then the model is repeatedly trained. The following graph of the dependence of the error on the length of the vector is obtained (see Fig. 3).

The error decreases sharply, reaches its global minimum, after which it slowly grows or remains at the same level.

Having repeatedly trained the model, the best quality of the model has been obtained. The minimum error is achieved with a vector length of 100.

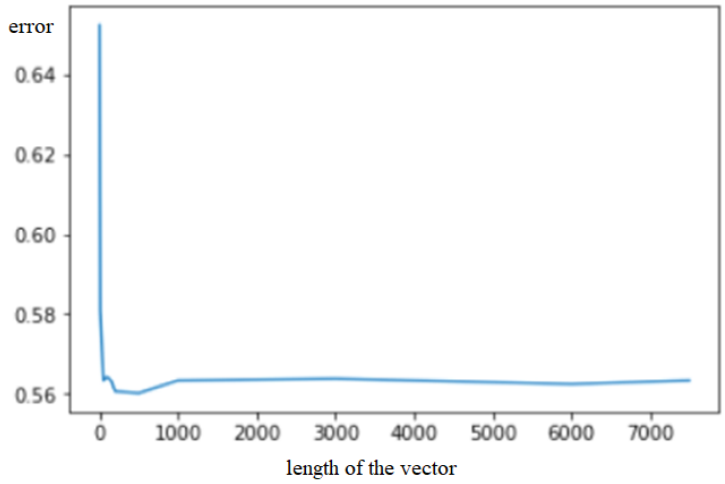

Fig. 3. Dependence of the error on the length of the vector.

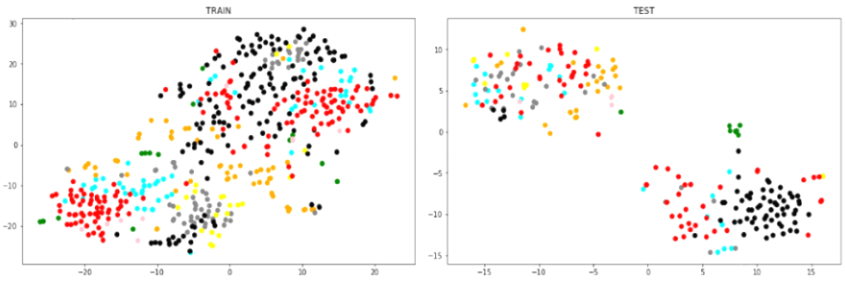

Fig. 4. Visualization graphs on the train and test data sets.

It is possible to select individual clusters if visualization has been created (see Fig. 4).

The data set is obtained in the form of 600 thousand news from the site Lenta.ru. After each training of the model, it is decided to manually measure the quality metric. A quality assessment is made better than in the case of books. In the experiment with books, one cluster is too large, that is why it was divided into 2-3 clusters automatically.

The following graph of the dependence of the error on the length of the vector is obtained (see Fig. 5).

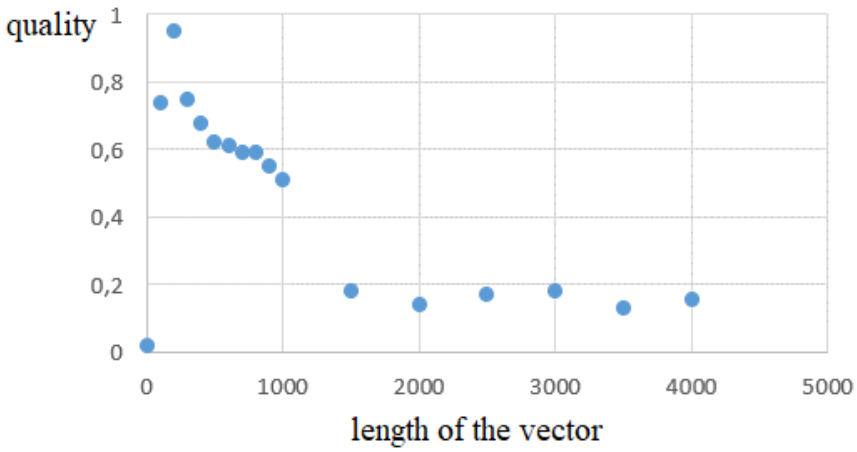

Fig. 5. Dependence of the quality on the length of the vector.

This approach gives an arithmetic average accuracy of about 90% when searching for similar news texts with the vector length of 200.

The quality of the model is getting better and better with increasing corpus text.

8 Summary

The following conclusions are made after the experiments:

- all classes should be presented in the training and test data sets.
- the algorithm does not work correctly on a small text corpus.
- train and test data sets should be large enough for the quality of the model to be good.
- vector representation allows determining the hidden (latent) meaning of texts based on the occurrence of words with each other. This technique gives an accuracy of about 90% when searching for similar news.
- the error decreases sharply, reaches its global minimum, slowly grows or remains at the same level.

9 Conclusion

The topic of this article is to search for similar documents using a vector representation of a document. It has been shown in practice how to find similar documents using doc2vec. The most popular methods for finding similar documents are discussed and the reasons for doc2vec to be used.. The operation of the Word2vec method and the doc2vec based on it are described.

Then the strategy is described, which allows obtaining the best quality of the doc2vec model. A series of experiments is carried out, which shows which data doc2vec reveals its potential on. The strengths and weaknesses of this approach are also identified.

It is found that the length of the document vector increases with increasing the body of texts. The quality of the model improves with increasing corpus text.

Until 2013, when Word2vec appeared, data scientists worked with a matrix of words in documents that is why the dimension of the matrix was high. Vector presentation of documents has reduced dimensionality, making the work with documents easier and more affordable.

The task of text classification, for which the logistic regression is mostly used, is not suitable for searching for similar documents, since dataset may not be known in advance. Taking some part of the sample, marking up, using doc2vec, calculating the arithmetic mean precision on the test sample is a more reasonable solution.

The LSA, PLSA, LDA algorithms are also used for tasks in which the word case is small, since the quality on them is better than that of doc2vec. Data is rapidly produced, and more and more data scientists are beginning to embrace doc2vec for such tasks.

References

1. Hilbert, P. Lopez: The Worlds Technological Capacity to Store, Communicate, and Compute Information. Science 332(6025), 60–65 (Oct. 2011).

2. Anastasia Chirkina, Marina Medvedeva, and Evgeny Komotskiy: CIN classification and prediction using machine learning methods. AIP Conf. Proc. 1836, 020010 (2017).
3. E. Agbozo: Developing a digital government framework for Sub-Saharan Africa. In: Proceedings of 17th European Conference on Digital Government, ECDG 2017, vol. F129463, pp. 294-305 (2017).
4. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean: Distributed representations of words and phrases and their compositionality. In: NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems, vol. 2, pp. 3111–3119 (2013).
5. J. Lilleberg, Y. Zhu, and Y. Zhang: Support vector machines and Word2vec for text classification with semantic features. In: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC (2015).
6. M.A. Medvedeva and M.A. Medvedev.: About the Use of Vector Optimization for Company's Contractors Selection, AIP Conference Proceedings 1863, 050011 (2017).
7. Q. Le and T. Mikolov: Distributed representations of sentences and documents. In: ICML'14 Proceedings, vol. 32, pp. 1188-1196 (2014).
8. M. Campr and K. Ježek: Comparing Semantic Models for Evaluating Automatic Document Summarization. In: Text, Speech, and Dialogue Lecture Notes in Computer Science, pp. 252–260 (2015).
9. E. F. Altszyler, M. F. Sigman, S. F. Ribeiro, and D. F. Slezak: Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. In: Conscious Cogn, pp. 178–187 (2017).