Feature Engineering Methods for Sentence Similarity
CUNY DATA-698 - MS in Data Science
Research Project

Olga Shiligin, 2020

Abstract

Feature Engineering is often known as the secret sauce to creating superior and better performing machine learning models. The importance of feature engineering is even more important for unstructured, textual data because we need to convert free flowing text into some numeric representations which can then be understood by machine learning algorithms. Even with the advent of automated feature engineering capabilities, we still need to understand the core concepts behind different feature engineering strategies before applying them as black box models.
This paper will demonstrate various approaches to feature engineering for sentence similarity tasks.

Keywords: sentence similarity, word embeddings, Smooth Inverse Frequency, Word Mover's Distance, gensim, word2vec.

---

1. Introduction

Today's machines can analyze more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data, automation will be critical to fully analyze text efficiently. As there is no inherent structure to text documents because we can have a wide variety of words which can vary across documents and each sentence will also be of variable length as compared to a fixed number of data dimensions in structured datasets. For that reason the importance of feature engineering in text similarity tasks is obvious [2].
This study will seek to explore various feature engineering techniques and their effectiveness for sentence similarity tasks by utilizing techniques from classical text mining features to more

advanced - word embeddings features, fuzzy features and exploring similarity measures on the word embeddings.

This research work uses Quora data set. Quora is an American question-and-answer social network where questions are asked, answered, and edited by users, either factually or in the form of opinions. Quora is a place to gain and share knowledge. It's a platform to ask questions and connect with people who contribute unique insights and quality answers. Millions of people use Quora every month, and many of them ask similar questions, so finding similar questions can significantly increase the efficiency of using this site.

Utilizing various feature engineering methods and building models we classified Quora questions on similar and not similar solving this way binary classification problem. Because the correct answer is associated with a group of questions regarded as similar, the accuracy of predictions for Quora is very important. Our focus in this study will be distributed as follows: text pre-processing, text data engineering, model building to assess the effectiveness of applied feature engineering techniques for sentence similarity tasks.

Tool set for this research includes Python packages for natural language processing like Gensim, NLTK, spacy, sklearn.

2. Literature Review

Measures of text similarity have been used for a long time in applications in natural language processing and related areas. The typical approach of finding the similarity between two text segments is to use a simple lexical matching method, and produce a similarity score based on the number of lexical units that occur in both input segments[4]. Improvements to this simple method have considered stemming, stop-word removal, part-of-speech tagging, longest subsequence matching, as well as various weighting and normalization factors [5]. While successful to a certain degree, these lexical matching similarity methods fail to identify the semantic similarity of texts.

One of the methodologies that have been used to determine similarity measures is the word overlap measure[15]. This method seeks to find related concepts through comparing the overlapping words between sentences of both concepts. In (Metzler, 2005)[16], a method of word overlap is evaluated from a simple word overlap fraction that determines the proportion of words that appear in the two sentences to compare them. Then this proportion is normalized through the length of the sentences. Another method based on word overlap used to determine similarity between sentences is the Inverse Document Frequency (IDF) overlap, where the proportion of words is compared in two sentences from their IDF weights. Based on these

ideas[17], extends the concept of word overlap to the distinction between multi-word terms. They assume that the simple word overlap methods do not take into account the elements that are composed of more than one word and that may be important for the similarity between sentences. Therefore, they estimate the overlap between multi-word terms and single word terms. Other methods are based on Term Frequency-Inverse Document Frequency (TFIDF). For example, (Allan, 2003)[18] is based on search thematically similar sentences. It is based on the sum of the TFIDF values of the words in both sentences, since it is assumed that this measure weights the thematically relevant words. Another way is to create vectors based on the idea of Bag of Words (BoW) and compare the distance between vectors sentences. However, BoW loses relevant information between the sentences, such as order, and have now been overtaken by the word embeddings representations [9]. BoW refers to the number of times that a word appears in a text. Feature selection through word frequency means to delete the words, whose frequencies are less than a certain threshold, to reduce the dimensionality of feature space. This method is based on such a hypothesis; words with small frequencies have little impact on filtration [10, 11]. However, in the studies of information retrieval, it is believed that sometimes words with less frequency of occurrences have more information. Therefore, it is inappropriate to delete a great number of words simply based on the word frequency in the process of feature selection [11].

Recent trends suggest that neural network-inspired word embedding models outperform traditional count-based distributional models on word similarity and analogy detection tasks.[13] Neural-network based approaches in which words are “embedded” into a low dimensional space were proposed by various authors (Bengio et al., 2003; Collobert and Weston, 2008). These models represent each word as a dimensional vector of real numbers, and vectors that are close to each other are shown to be semantically related. It was popularized via word2vec, a program for creating word embeddings.

For a sentence embedding method Smooth Inverse Frequency (SIF) was introduced where the weighted average of the word vectors in the sentence is computed and then the projections of the average vectors on their first singular vector (“common component removal”) is removed. This method achieves significantly better performance than the unweighted average on a variety of textual similarity tasks, and on most of these tasks even beats some sophisticated supervised methods tested in (Wieting et al., 2016), including some RNN and LSTM models. The method is well-suited for domain adaptation settings, i.e., word vectors trained on various kinds of corpora are used for computing the sentence embeddings in different testbeds. It is also fairly robust to the weighting scheme: using the word frequencies estimated from different corpora does not harm the performance; a wide range of the parameters can achieve close-to-best results, and an even wider range can achieve significant improvement over unweighted average[12].

Fuzzy logic has been successfully applied to the description of words’ meanings as related to language external phenomena. Fuzzy linguistic descriptors have been used in control systems, in which mappings can be established between fuzzy linguistic terms and physical quantities. In linguistic research there has always been a tendency to treat linguistic categories and structures

as fuzzy entities. This is strongly reflected in the cognitive grammar tradition. In that tradition, prototypes of natural categories and lexical semantics are considered as fuzzy and gradient in membership assignment [14].

3. Methods

3.1 Data

In order to assess the performance of various feature engineering methods Quora data set was taken. It consists of 404268 observations and 6 columns. The columns are the following: question pair id, unique ids of each question, the full text of each question and target variable "is_duplicate". Target variable set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise. Data set consists of 37% of duplicate and 63% of unique question pairs and this indicates that it is moderately unbalanced.

Wikipedia word frequency list was also used and it was assisting in calculating Smoth Inverse Frequency (SIF). File contains 2 184 780 words and their frequencies.

3.2 Strategy

Initially we have balanced data set via downsampling reducing it to 100 k observations with equal representation of each class. After a number of pre-processing steps the data set was prepared for feature engineering. This paper assesses the following feature engineering approaches for sentence similarity: classical text mining features - basic and advanced, fuzzy features, bag of words methods, word embeddings features (Word2Vec) and similarities calculated based on it, Smooth Inverse Frequency (SIF) and Word Mover's Distance (WMD). For comparison reason the same models were employed on the generated features which allowed us to select the best feature engineering methods for our sentence similarity task.

3.3 Methods

3.3.1 Data Pre-Processing

Text cleaning is an essential pre step of any NLP task. In order to perform these computational tasks, we first need to convert the language of text into a language that the machine can understand. The following steps were taken to prepare data for feature engineering process:

Normalization. One of the key steps in processing language data is to remove noise so that the machine can more easily detect the patterns in the data. Text data contains a lot of noise, this takes the form of special characters such as hashtags, punctuation and numbers, all of which are

difficult for computers to understand if they are present in the data. We have processed the data and removed or transformed the elements such as punctuation, special characters, abbreviations, floating numbers and money signs.

Stop words. Stop words are commonly occurring words that for some computational processes provide little information or in some cases introduce unnecessary noise and therefore need to be removed[2]. This is particularly the case for sentence similarity tasks. There are some instances where the removal of stop words is either not advised or needs to be more carefully considered. This includes any situation where the meaning of a piece of text may be lost by the removal of a stop word. Taking this into consideration we performed some feature engineering methods with and without stop words filtering.

Stemming. Stemming is the process of reducing words to their root form and this a way to reduce noise and the dimensionality of the data. The problem with stemming is that it can lead to the losing the meaning of the sentence. As a better alternative to stemming lemmatization was considered. This is the same as for stemming, in that it aims to reduce words to their root form. However, stemming is known to be a fairly crude method of doing this. Lemmatization, on the other hand, is a tool that performs full morphological analysis to more accurately find the root, or "lemma" for a word[19].

3.3.2 Feature Engineering Methods

For robust natural language processing to succeed we have considered and implemented various feature engineering methods. We have started from the most simple ones, because whether constructing machine learning models or engineering features, it's good when the results are simple and interpretable.

Basic Features

We started from basic features like number of words in each sentence, question length (number of characters), average token length of both questions, number of common unique words and their share in Question 1 and Question 2 etc. Utilizing basic NLP feature creation techniques we have created 14 features. List of basic features presented below:

Number of words in Question 1 and 2
Length of Question 1 and Question 2
Number of common words in questions
Share of common words in questions

Number of common words to min(max) length of word(token, stop words) count of Q1 and Q2
Token length difference of Question 1 and Question 2

Fuzzy logic based features

Fuzzy logic is a form of multi-valued logic that deals with reasoning that is approximate rather than fixed and exact(True/False). Fuzzy logic values range between 1 and 0 and gives more flexibility than True/False approach[20]. Fuzzy String Matching, also known as approximate string matching, is the process of finding strings that approximately match a pattern. We used Python library Fuzzywuzzy which calculates the differences between sequences and patterns utilizing Levenshtein Distance. There are several ways to compare two strings in Fuzzywuzzy, we have tried them one by one in our research project. Ratio compares the entire string similarity in order. It turns out, the naive approach is far too sensitive to minor differences in word order, missings or extra words, and other such issues. Partial ratio compares partial string similarity, token sort ratio ignores word order, while token set ratio ignores duplicated words, it is similar with token sort ratio, but a little bit more flexible.

Bag Of Words

Our further focus was directed on bag of words feature techniques such as Bag of Words (BoW) and TF-IDF featurization methods. Implementation of BoW featurization has allowed us to convert sentences into a vector of counts. The vector contains an entry for every possible word in question 1 and question 2. If a word in the vocabulary doesn't appear in the document, then it gets a count of 0. Bag-of-words converts a text document into a flat vector. It is "flat" because it doesn't contain any of the original textual structures. The original text is a sequence of words. But a bag-of-words has no sequence, it just remembers how many times each word appears in the text.[2] Considering the limitation of BoW featurization method we implemented TF-IDF method. It stands for term frequency– inverse document frequency. Instead of looking at the raw counts of each word in each document in a dataset, TF-IDF looks at a normalized count where each word count is divided by the number of documents this word appears in. TF-IDF makes rare words more prominent and effectively ignores common words.
We have constructed TF-IDF vectors at different levels of input tokens: word level, where matrix represents tf-idf scores of every term, n-gram Level - every n-gram term and character level.

Word Embeddings

Another type of featurization methods that were tested is word embeddings method. Word embedding — the mapping of words into numerical vector spaces — has proved to be an incredibly important method for natural language processing (NLP) tasks in recent years,

enabling various machine learning models that rely on vector representation as input to enjoy richer representations of text input[9]. When it comes to raw text data, especially count based models like Bag of Words, we are dealing with individual words which may have their own identifiers and do not capture the semantic relationship amongst words. This leads to huge sparse word vectors for textual data and thus if we do not have enough data, we may end up getting poor models or even overfitting the data due to the curse of dimensionality. Predictive methods like Neural Network based language models try to predict words from its neighboring words looking at word sequences in the corpus and in the process it learns distributed representations giving us dense word embeddings. Such models are based on distributional hypothesis which is derived from the semantic theory of language usage, i.e. words that are used and occur in the same contexts tend to purport similar meanings[21].

We have implemented one of word embedding models - Word2vec. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.

There are several pre-trained word2vec models that are available to implement. We used Google's pre-trained model. It includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features for each question.

Similarity Metrics

Results of word2vec implementation were also utilized using Word Mover's Distance (WMD), cosine similarity and Manhattan distance. WMD is designed to overcome synonym problem. WMD uses word embeddings to calculate the distance so that it can calculate even though there is no common word. It measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. The assumption is that similar words should have similar vectors. WMD is far more expensive to calculate, especially on longer texts. The steps to calculate WMD are the following: retrieve vectors from any pre-trained word embeddings models, after that it uses normalized bag-of-words (nBOW) to represent the weight or importance. It assumes that higher frequency implies that it is more important.

We also calculated cosine similarity, which is calculated as 1 - cosine distance. Cosine distance is the distance between two vectors in n dimension space and represents how words are related to each other. Taking the average of the word embeddings in a sentence is a very crude method of computing sentence embeddings. Most importantly, this gives far too much weight to words that are quite irrelevant, semantically speaking. Smooth Inverse Frequency tries to solve this

problem. To compute SIF sentence embeddings, we first compute a weighted average of the token embeddings in the sentence. This procedure is very similar to the weighted average we used above, with the single difference that the word embeddings are weighted by

$$a/a+p(w),$$

where w is a parameter that is set to 0.001 by default, and p(w) is the estimated relative frequency of a word in a reference corpus (wikipedia word frequency corpus). Next we needed to perform common component removal: we compute the principal component of the sentence embeddings we obtained above and subtract from them their projections on this first principal component [12].

3.3.3 Performance Metrics

To evaluate significance of the derived features, we have built 3 types of models: Logistic Regression, Random Forest and XGBoost on each feature set/method. We have trained our algorithms on training data set and obtained accuracy using 3-fold cross validation. We have selected the algorithm with the highest accuracy and then applied it on the test data set.
F1 score and Recall were used as a performance metrics. In our research the correct answer is based on a pair of questions regarded as similar. In order to achieve best accuracy in classifying duplicate questions we focused on Recall as it shows that how many of truly duplicate questions we labelled correctly. We also considered accuracy as we have balanced our data set by under-sampling.

4. Results

Results of the research are presented in Table 1. Word embedding method based on word2vec is the feature engineering method that allowed us to get the most accurate prediction of the duplicate question pairs (accuracy and recall - 0.89). At the same time less complex method like TF-IDF with character analyser made relatively good predictions as well - accuracy and recall are 0.8 and 0.81 respectively, while TF-IDF with n-grams of order 2 and 3 showed the worst performance - accuracy 0.69 and recall 0.63. Basic features set as well as features derived based on Fuzzy logic showed high recall - 0.85 and 0.86 respectively, indicating that of all questions that are duplicate 86% we labelled correctly. Various similarity metrics, calculated using word2vec vectors did not give impressive results with accuracy - 0.72 and recall - 0.8. Smooth Inverse Frequency method just slightly increased the accuracy and recall.

Features+Model	accuracy	F1	recall	precision
Basic				
Basic + XGBoost	0.75	0.77	0.85	0.71
Basic+Fuzzy+ XGBoost	0.77	0.79	0.86	0.73
Bag Of Words				
BoW(word) + XGBoost	0.77	0.77	0.77	0.76
BoW(char) + XGBoost	0.77	0.77	0.79	0.75
TF-IDF(word) + XGBoost	0.78	0.78	0.79	0.77
TF-IDF(n-gram) + XGBoost	0.69	0.67	0.63	0.72
TF-IDF(char) + XGBoost	0.8	0.8	0.81	0.8
Word Embeddings				
Word2Vec + XGBoost	0.89	0.9	0.89	0.9
Similarity metrics				
wmd+norm_wmd+cosine+XGBoost	0.72	0.74	0.8	0.69
wmd+norm_wmd+cosine+SIF+XGBoost	0.73	0.75	0.81	0.7
Basic+Fuzzy+ Word Embeddings + XGBoost	0.76	0.77	0.83	0.71

Table 1: Performance of the Feature Engineering Methods

5. Conclusion

This research paper has demonstrated various approaches to feature engineering in order to solve sentence similarity task from deriving basic features to more advanced methods. The effectiveness of the feature engineering in sentence similarity analysis demonstrated direct and significant influence on the accuracy. Word embedding method showed best results on our non-domain specific data set. This method was developed on Word2Vec model pre-trained on the very large news data set and showed good results identifying sentence content.

TF-IDF method is not as powerful as word embedding method, but still gave good accuracy and recall and can be used as building baseline feature engineering approach.

Bag of Words (BoW) failed to produce good results with given data set as it is . This was expected as this approach is less preferred method for non-domain specific compare to word embeddings.

Derived basic features like number of word, length of questions, common word share, token length difference etc. together with Fuzzy logic based features showed very high recall and can be used as a starting point or even compete with bag of words methods.

The results of implementing Smooth Inverse Frequency did not justify the complexity of its computation as it could not improve the class prediction accuracy substantially. Similarity metrics calculated based on word2vec results such as cosine similarity, Manhattan distance, Word Mover's Distance did not show significantly higher accuracy or recall compare to bag of words methods.

While this research paper covered major types of feature engineering methods for sentence similarity tasks, graph-based feature engineering methods were not considered here and can be a good topic for further research.

6. Appendix

All code for this project can be found at:

https://github.com/olgashiligin/data_698

7. References

[1] - Wael H. Gomaa, Aly A. Fahmy, A Survey of Text Similarity Approaches, International Journal of Computer Applications (0975 – 8887), 2013

[2] - Alice Zheng & Amanda Casari, Feature Engineering for Machine Learning: Principles ans techniques for data scientists, 2018.

[3] - Voorhees. 1993. Using wordnet to disambiguate word senses for text retrieval. In Proceedings of the 16th annual international ACM SIGIR conference, Pittsburgh, PA.

[4] - Salton and M.E. Lesk. Computer evaluation of indexing and text processing. Prentice Hall, Ing. Englewood Cliffs, New Jersey, 1971

[5] - Salton and A. Bukley. Term weighting approaches in automatic text retrieval. In Readings in Information Retrieval. Morgan Kaufmann Publishers, San Francisco, CA, 1997

[6]- K. Landauer, P. Foltz, and D. Laham. Introduction to latent semantic analysis, 1998.

[7] - 3. Singh V, Kumar B, Patnaik T. Feature extraction techniques for handwritten text in various scripts: a survey. International Journal of Soft Computing and Engineering. 2013;3(1):238–241. [Google Scholar]

[8] - 10. D Mladenic, Machine learning on non-homogeneous, distributed text data, PhD Thesis. Web. (1998)

[9] - Victor Mijangos, Gerardo Sierra and Abel Herrera. A Word Embeddings Model for Sentence Similarity. Research in Computing Science 117 (2016)

[10] - 11. S Niharika, VS Latha, DR Lavanya, A survey on text categorization. Int. J. Compt. Trends Technol. 3(1), 39-45 (2006)

[11] - 12. Mhashi M, Rada R, Mili H, et al, Word Frequency Based Indexing and Authoring[M]// Computers and Writing. (Springer, Netherlands, 1992), p. 131-148.

[12] - Sanjeev Arora, Yingyu Liang, Tengyu Ma, A simple but tough-to-beat baseline for sentence embeddings. Princeton University, a conference paper at ICLR 2017

[13] - Omer Levy, Yoav Goldberg, Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings

[14] - Jiping Sun, Fakhri Karray, Otman Basir & Mohamed Kamel. Fuzzy Logic-Based Natural Language Processing and Its Application to Speech Recognition. 2002

[15] - Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The Evaluation of Sentence Similarity Measures, College of Information Science and Technology, 2008

[16] - Donald Metzler, Christopher Meek. Similarity Measures for Short Segments of Text.Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007.

[17] - Satanjeev Banerjee, Ted Pedersen. Extended Gloss Overlaps as a Measure of Semantic Relatedness.Carnegie Mellon UniversityPittsburgh, 2003.

[18] - Juan Enrique Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. Department of Computer Science, Rutgers University, 2003

[19] - Tuomo Korenius, Jorma Laurikkala, Kalervo Järvelin, Martti Juhola. Stemming and Lemmatization in the Clustering of Finnish, Department of Computer Sciences, Center for Advanced Studies University of Tampere, Finland

[20] - Fuzzy Systems: Concepts, Methodologies, Tools, and Applications, Management Association, Information, 2017.

[21] - The distributional hypothesis. Italian Journal of Linguistics, 2008