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

Olga Shiligin, 2020

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.

Literature Review

Measures of text similarity have been used for a long time in applications in natural language processing and related areas.

The very basic 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

Recent trends suggest that neural network-inspired word embedding models outperform traditional count-based distributional models on word similarity and analogy detection tasks.

Sources of literature: Academic Paper, Scientific Conference reports, Books

The Dataset

404268 observations

6 columns: 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 question 1 and question 2 have essentially the same meaning, and 0 otherwise.

Data set consists of 37% of duplicate and 63% of unique question pairs

Data set was balanced via downsampling and splitted into training and validation data sets (70/30)

Project Purpose

The purpose of the project is to explore various approaches to feature engineering sentence similarity analysis.

Approach

After a number of pre-processing steps the data set was prepared for feature engineering. This research explores the following feature engineering approaches: 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).
Same models were applied on the generated features. This allowed us to select the best feature engineering methods for our sentence similarity task.

Basic Features

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 words(token, stop words) of Q1 and Q2
Token length difference of Question 1 and Question 2

Fuzzy Logic Based Features

The following fuzzy logic based ratios were calculated:

- fuzzy ratio
- partial ratio
- token sort ratio
- token set ratio

The ratios were calculated using Python *Fuzzywuzzy* and *Levenshtein* packages.

Bag of Words Methods

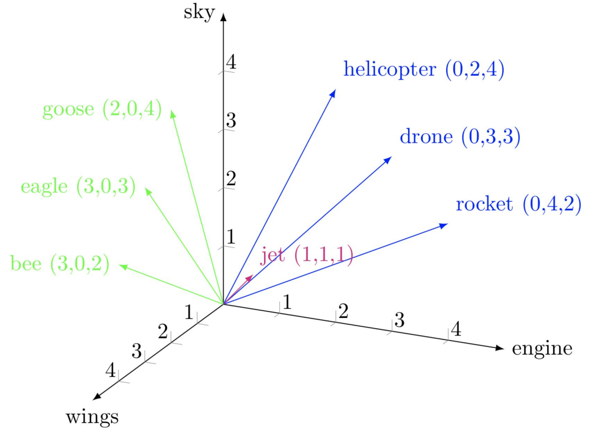
Bag of Words - raw counts of each word in each sentence

TF-IDF - normalized count where each word count divided by the number of sentences this words appear in.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Word Embeddings



- the mapping of words into numerical vector spaces;
- Neural Network based language models;
- based on distributional hypothesis: words that are used and occur in the same contexts tend to purport similar meanings

Word2Vec

- shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.

- 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.

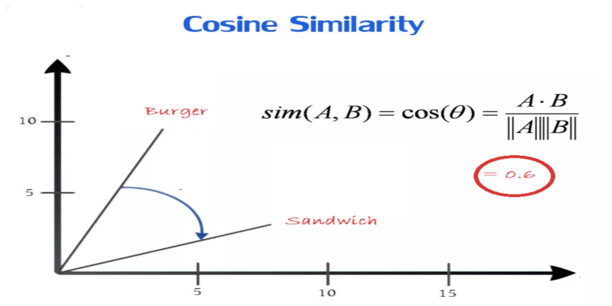
Word2Vec

- 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



Cosine distance is the distance between two vectors in n-dimension space and represents how words are related to each other.

cosine similarity = 1 - cosine distance

Manhattan distance is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates.

Word Mover’s Distance (WMD)

- designed to overcome synonym problem
- uses word embeddings
- calculate the distance so that it can calculate even though there is no common word
- 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.

Smooth Inverse Frequency (SIF)

- taking the average of the word embeddings in a sentence is a very crude method of computing sentence embeddings and SIF tries to solve this issue.

- we first compute a weighted average of the token embeddings in the sentence:

  a/a+p(w), where w is a parameter that is set to 0.001 by default,
  p(w) is the estimated relative frequency of a word in a reference corpus .

- 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.

- wikipedia word frequency corpus was used

Results

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	0.74	0.75	0.81	0.7
Basic+Fuzzy+ Word Embeddings + XGBoost	0.76	0.77	0.83	0.71

Conclusion

- Word embeddings method outperformed other feature engineering methods on non-domain specific sentence similarity analysis

- 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.

- Derived basic features like and 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 measures and distances calculated based on word2vec results such as cosine, Manhattan distance, Word Mover's Distance did not show significantly higher accuracy or recall compare to bag of words methods.

- Bag of Words (BoW) failed to produce good results with given data set as it usually weaker method for non-domain specific compare to word embeddings.