Identifying duplicate Quora Question pairs

Barleen Kaur
Masters in Computer Sciences
McGill University
barleen.kaur@mail.mcgill.ca

Sihyeon Kim
Department of Arts
McGill University
sihyeon.kim@mail.mcgill.ca

Abstract

Sentence similarity measures have proven to be vital in text-related research and applications. Since most of the text classification methods need text to be represented in vector form , we evaluate the performance of various vectorization techniques (n-grams, word2vec and doc2vec) on several shallow classifiers. For this, we have considered the publically available Quora dataset which consists of pairs of questions. The real interest behind the challenge is to classify whether these question pairs are duplicates or not. Since Quora is one of the most popular Question-answer platforms for information exchange and has over 190 million visitors every month, solving this challenge helps us to identify questions having the similar intent which can later be removed, making it easier to filter high-quality answers to questions and resulting in an improved experience for Quora writers, seekers, and readers.

1 Introduction

Quora[1] is a website where questions can be asked, answered, shared and organized by its community of users. As of December 2017, there are over 21 million questions posted. Compared to the beginning of this year, which had around 12 million, the amount of questions almost doubled, and this shows that more and more people are using this Q&A community. The noticeable problem from which both users and the website are suffering is that many of the questions are asked repeatedly. For the website, a lot of data space is wasted over the same topics, which led them to increase their database maintenance cost. For the users, they waste time on writing repetitive questions to get duplicate answers. If Quora designed the system providing "possible" duplicated questions before the user asks, then it would be beneficial for the website and the user experience. Since multiple users can generate questions with same intent in different ways, it would be necessary to detect the questions holding the same semantic meaning but using different words and/or phrases. There are many techniques to find the duplication of questions. The most common practice is text classification and clustering. It applies with machine learning algorithms such as Support Vector Machine (SVM) or logistic regression which require the text input to be represented as a fixed-length vector. One of the most common fixed-length vector representation for a text is called bag-of-words or n-gram model. This model focuses on finding whether the words in two sentences are arranged in similar order and with similar vocabulary. It is cost-efficient, simple, and results in quite nice turnout. However, it fails to catch the semantic meaning of the sentences because it looks only at the relationship between sentences at the individual word-level. For example, if there are two sentences *"Will there be a world war 3 soon? Why or why not?"* and *"how close to ww3 are we?"*, even though it has the same semantic meaning with similar intent, those methods would fail to catch the similarity. On the other hand, there have been several techniques which capture semantics features. It is argued that the results of those methods significantly outperform the one of n-gram model.

In this paper, we used the Quora duplicate questions public dataset [2] to show the importance of semantic relatedness using distributed representation of words model.Our model needs to recognize even when two questions which are written in different ways, but have the same semantic meaning, would be signed as "duplicate questions".

[1] Website: https://www.quora.com/
[2] Quora Dataset

2 Related Work

One of the most common fixed-length vector representations of text is the bag-of-words (or bag-of-n gram) model Harris (1954). It is popular for providing a relatively good performance given its cost to run. However, it has several limitations such as losing the word order, representation in high dimension, and sparsity in the representation. Furthermore, this model is not able to capture the semantics of the sentences. To address the limitations of n-gram language model, several distributional semantics models were introduced, which were based on the notion that the words appearing in the similar context will contain similar meaning. One of the first models was called Latent Semantic Analysis, or LSA Deerwester et al. (1990). LSA builds a word-document co-occurrence matrix and performs singular value decomposition. The limitation of the LSA is that, for large corpus, the co-occurrence matrix can have a significant size. Mikolov et al. (2013) introduced Word2Vec model, which complements the limitations of the LSA model. It focuses on continuous space models, where similar words are likely to have similar vectors, learned by neural networks. It significantly outperforms the LSA by preserving linear regularity among words at a cheaper cost. While traditional approaches predicted a word conditioned on its predecessor, word2vec uses two methods, a continuous bag of words (CBOW) and continuous skip gram.
Further research deepened these perspectives by introducing sentence or even document (text) level semantic analysis. Le and Mikolov (2014) proposed an adaptation of the word2vec algorithm to calculate paragraph vectors. Every paragraph is mapped to a unique vector, and every word is also mapped to a unique vector. The paragraph and word vectors are averaged or concatenated to predict the next word in the context. It is an unsupervised framework that learns continuous distributed vector representations for pieces of texts.

3 Understanding the Quora Dataset

Quora released its first public dataset in January 2017 in order to solve the problem of identifying duplicate question pairs. This was done to keep each logically and semantically unique question in a single location so that users can access all answers related to that question in the same page. For example: questions like *"what is the best book to read to learn java?"* and *"which is best book for java?"* should be clubbed together for better user readability as they share the same semantics.

Fields	Description
id	unique id of the question pair
qid1	unique id for each question1
qid2	unique id for each question2
question1	full text of question1
question2	full text of question2
is_duplicate	$\{0, 1\}$ 1 if duplicate

Table 1: Structure of Quora Dataset

Type	Count
Duplicate questions	$149,263$
Non-duplicate questions	$255,027$
Unique questions	$537,933$
Multiple occurrences of questions	$111,780$
Empty/missing questions	2

Table 2: Table describing the preliminary analysis of the data

3.1 Data Analysis

The quora dataset consists of 404,290 question pairs in the structure as described in Table 1. The dataset contains 149,263 duplicates and 255,027 non-duplicate questions which comprise of approximately 36.91% and 63.08% of the full dataset respectively as described in Table 2.

While analysing the dataset, we looked at the distribution of words in the sentence and found that the number of words in a question can range from 1 to 237 as shown in Figure 1. On close examination, we also found that there are few questions in the dataset which are semantically close but distinct, still they are not labelled as duplicate. Also, negative samples introduced by noisy human curated labels make the dataset ambiguous. For example: *"Will there be a nuclear war between India and Pakistan?"* and *"If war happens between India and Pakistan, how will it affect common man economically?"* are labelled as duplicates, but *"Is Justin Trudeau a good leader for Canada in your opinion so far?"* and *"How good a prime minister of Canada is Justin Trudeau?"* are not.

4 Methodology

In order to identify the duplicate question pairs, we divided the task into three stages: Preprocessing

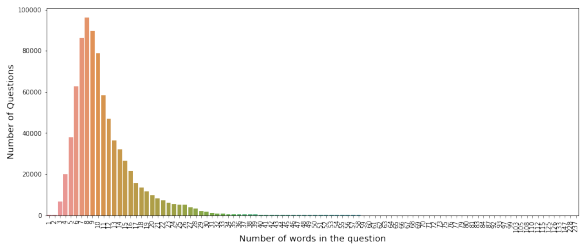

Figure 1: Distribution of words in the questions

the text, generating the embedding and passing the embedding into shallow classifiers for prediction.

4.1 Preprocessing

The preprocessing of the dataset was done in three ways:

Type I: In this approach, we converted the text into lowercase and removed all the non-alphanumeric characters along with the punctuations.

Type II: In this approach, we converted the text into lowercase. While doing data analysis, we noticed that multiple word forms were used to represent the same word. For example: "kms", "km" for *"kilometer"*; "U.S.", "US", "U.S.A" for *"America"*; "III" for *"3"*; "cs" for *"computer science"*. So, we manually corrected all these forms to a single form across the whole dataset for disambiguation. We removed all non-alphanumeric characters and punctuations along with the stopwords present in nltk's [3]corpus package. The text was then tokenized into words from which every token was lemmatized and joined together to form a cleaned sentence.

Type III: This approach differs from Type 2 in terms of removing stopwords from only a selective list which is a subset of nltk's stop words list. This was done to include words like "not", "down", "against", "few" etc in the text which highly impact the semantics of the sentence. For example, consider a question *"what is the greatest thing money can't buy?"*, if the word *"can't"* is removed from the question, the polarity of the question becomes complete opposite which could impact the prediction of duplicity badly.

4.2 Feature Extraction

Since the questions are of varying length, representing the textual data in vector space model is necessary so that every question has the same dimensionality. For this, we used three standard NLP feature extraction techniques: N-grams (unigrams, bigrams, uni-bigrams), word2vec embedding and doc2vec model as described below:

N-gram Type	Unigram	Bigram	UniBigram
Type 1	9701	44206	53907
Type 2	8529	31163	39692
Type 3	8618	38807	47425

Table 3: Number of n-gram features per type

N-grams: We generated n-grams for both the questions using TfidfVectorizer[4]present in sklearn [5]library. Term frequency-inverse document frequency weighs term frequencies in the document by removing the meaningless words from the corpus. The distribution of n-gram features extracted per preprocessing type is shown in Table 3. While building the vocabulary, we included every term. The vectors generated for both the questions were then concatenated.

Word embeddings: We extracted the 300-dimensional vector representation of words in the text from Google's [6]pretrained word2vec model. We took the sum and mean vector of the word embedding of all the tokens in the sentence. For cases where word2vec model didn't have a word embedding or we encountered an empty question, we generated a random dense vector of 300 dimensions.

Doc2vec: We inferred a 300-dimensional continuous valued vector for every sentence using pretrained English Wikipedia [7]DBOW model with a step size of 1000.

4.3 Experimental Design

Experiment I: In the first experiment setting, we sampled 50000 train and 5000 test question pairs from the original dataset in such a way that there was equal distribution of duplicate and non-duplicate questions in both. We preprocessed them as described in Section 4.1, and generated n-grams, word embedding and paragraph vector for

[3] Website: http://www.nltk.org/

[5]Website: Tfidf link

[5]Website: http://scikit-learn.org/stable/

[6]Website:https://code.google.com/archive/p/word2vec/

[7]Website:https://ibm.ent.box.com/s/3f160t4xpuya9an935k84ig465gvymm2

Type-1						
Embedding style	word2vec		doc2vec	Bag of words		
Classifier	averaging	summing		unigram	bigram	uni+bi
SVM	47.6	49	51.2	65	66.4	68.2
Logistic Regression	48.8	48.4	51.2	68.8	65.6	67.2
Naïve-Bayes'	47.8	49	47.8	66.2	63.4	67.2
cosine	62.2	62.2	71.6	70.4	61.6	66.2

Type-2						
Embedding style	word2vec		doc2vec	Bag of words		
Classifier	averaging	summing		unigram	bigram	uni+bi
SVM	52.6	51.6	51	64.8	64.6	67.6
Logistic Regression	52	52.2	51.6	68.2	63.6	67.2
Naïve-Bayes'	53.2	50.8	51.2	63.8	61.6	63.8
Cosine	69.8	69.8	72.4	70	60.2	66

Type-3						
Embedding style	word2vec		doc2vec	Bag of words		
Classifier	averaging	summing		unigram	bigram	uni+bi
SVM	51.2	48.2	47.4	67.2	66.4	70.4
Logistic Regression	50	48.4	47.2	67.6	66.4	70.2
Naïve-Bayes'	51.6	49	48.4	63	62	65
Cosine	70.8	70.8	72	69.4	59.8	65

Table 4: Accuracy obtained for three different types of analysis for Experiment III

the question pairs. However, due to lack of computational resources, training the classifiers was a time-consuming process, so, we had to shorten our dataset size.

Experiment II: For the second experiment, we sampled 5000 train and 500 test questions from our original dataset keeping equal distribution of duplicate and non-duplicate questions. We generated all the ngrams, word2vec embedding. The paragraph vectors were produced by training the doc2vec model on our training dataset. All the feature vectors after normalization were passed into three shallow classifiers: Naïve Bayes, logistic regression, Linear SVM. We then, performed a five-fold cross validation based on variety of parameters such as cost, kernel, regularization. In addition to this, we also computed the cosine similarity scores between the vectors and found an optimal threshold using a cross validated dataset.

Experiment III: This experiment setting differs from experiment 2 only in terms of generating paragraph vectors from an already pretrained Wikipedia doc2vec model.

5 Results

In terms of classification accuracy, bag of words is performing better than word2vec and doc2vec as shown in analysis table 4. With respect to cosine similarity scores, word2vec and doc2vec outperform the baseline bag of words model and give an accuracy that is slightly higher than the shallow classifiers. Type 3 pre-processing performs significantly better than other types of pre-processing for bag of words embedding. But for word2vec and doc2vec models, additional pre-processing did not help in the performance.

6 Discussion

We observe from results that simple approaches such as summing and averaging for word2vec don't provide good results. Further work has to be done to develop sophisticated approaches for combining the words in word2vec models. Secondly, pre-trained doc2vec trained on Wikipedia corpus is able to generate similar words for any input of words. In spite of this, doc2vec is not able to capture the subtle differences between the two question pairs. This proves that the pre-trained models has to be fine-tuned to our datasets before making a prediction/classification. In addition, shallow classifiers such as SVM, Naive Bayes, and Logistic regression are not powerful enough to capture the subtle differences between the two questions. We may want to consider classifiers such as Neural Networks, LSTMs, and RNNs. Also, co-

sine distances are able to capture the similarities between the questions well. We also notice that pre-processing from one of the package libraries hinders the performance to some extent. Manual, data-specific pre-processing will help us to achieve a better results. The inherent noise in the Quora dataset as described in Section 3.1 is also responsible for decreased accuracy. Morever, including auto-correction of the misspellings present in the dataset can also help improve the performance of the three models.

7 Conclusion

In this work we experimented several pre-processing schemes with various embedding techniques on publicly available Quora dataset. We trained three different classifiers to measure accuracy. We notice that a better pre-processing scheme will yield a better accuracy. Also, we observe that fine-tuning is required for pre-trained models. Additionally, n-grams are able to capture the semantic context in the sentence and has a higher accuracy compared to the other two embedding styles.

8 Statement of Contribution

Barleen Kaur designed the overall structure and methodology of this project. She contributed in preprocessing of the data, performing data analysis, implementing word2vec, doc2vec models, and some parts of bag-of-words. She performed experimentation on three classifiers for the above mentioned embedding styles, produced cosine similarity scores, and contributed for the analysis of the results. She is also responsible for writing the abstract, methodology, experimentation, results and conclusion sections of the report.

Sihyeon Kim mainly participated in analyzing the dataset and implementing the code for the baseline method, the bag-of-words model, with three different classifiers. He conducted the part of the experimentation of bag-of-words on three classifiers. After the experiment was done, he contributed to the analysis and discussion of the results. In terms of writing this report, he was responsible for the introduction and related work (background) section.

References

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1188–1196.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.