***Finding Duplicate Quora Question Pairs***

Antriksh Agarwal

Department of Computer Science

The University of Texas at Dallas

Email: antriksh.agarwal1@utdallas.edu

***Abstract*—This project aims to come up with a system to identify pairs of questions of intention equivalence. The project identifies the specific distribution information of the data set adhering to how questions are asked on Quora. Then, some popular methods of representing sentences, and how these could be used in the context of the data at hand, are investigated. All through the project, a measure of similarity is chosen, which coincides with the model being built for representing the sentences. Three different techniques of representing sentences to measure similarity are implemented and scrutinized. The resulting performances are compared on logarithmic loss of similarity predictions of the three techniques.**

***Keywords—sentence similarity; intent; cosine similarity; tfidf; doc2vec; semantic similarity***

# Introduction

Where else but Quora can a physicist help a chef with a math problem and get cooking tips in return? Quora is a place to gain and share knowledge—about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

This project aims identify questions, asked on Quora, which have similar intent. Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.

This could be useful, for example, to instantly provide answers to questions that have already been answered. We are tasked with predicting whether a pair of questions are duplicates or not, and submitting a binary prediction against the log-loss metric.

# Related Work

Previous work has been done for similarity measurement of text, be it bag-of-words models or complex semantic systems. Reference [7] enumerates over some word overlap measures, TF-IDF measures, and linguistic measures that have been used for sentential similarity.

There have also been various attempts, like [6], to combine some techniques of extraction of semantic relations of in text in order to be able to better measure similarity. Some others [5] have used bipartite graph methods to try to compare two sentences and evaluate sentence similarity. This project uses some of these works as reference and looks for a basic approach to designate similarity to sentences.

# Proposed Approach

## *Flow of Proposed System*

This implementation makes use of Natural Language Toolkit(NLTK) [4] and genism [10] libraries in Python to perform various relations extractions in order to measure similarity between the questions as obtained from the dataset. Fig. 1 shows a high-level architecture of the proposed approach.

rchitecture.png

1. High Level Architecture

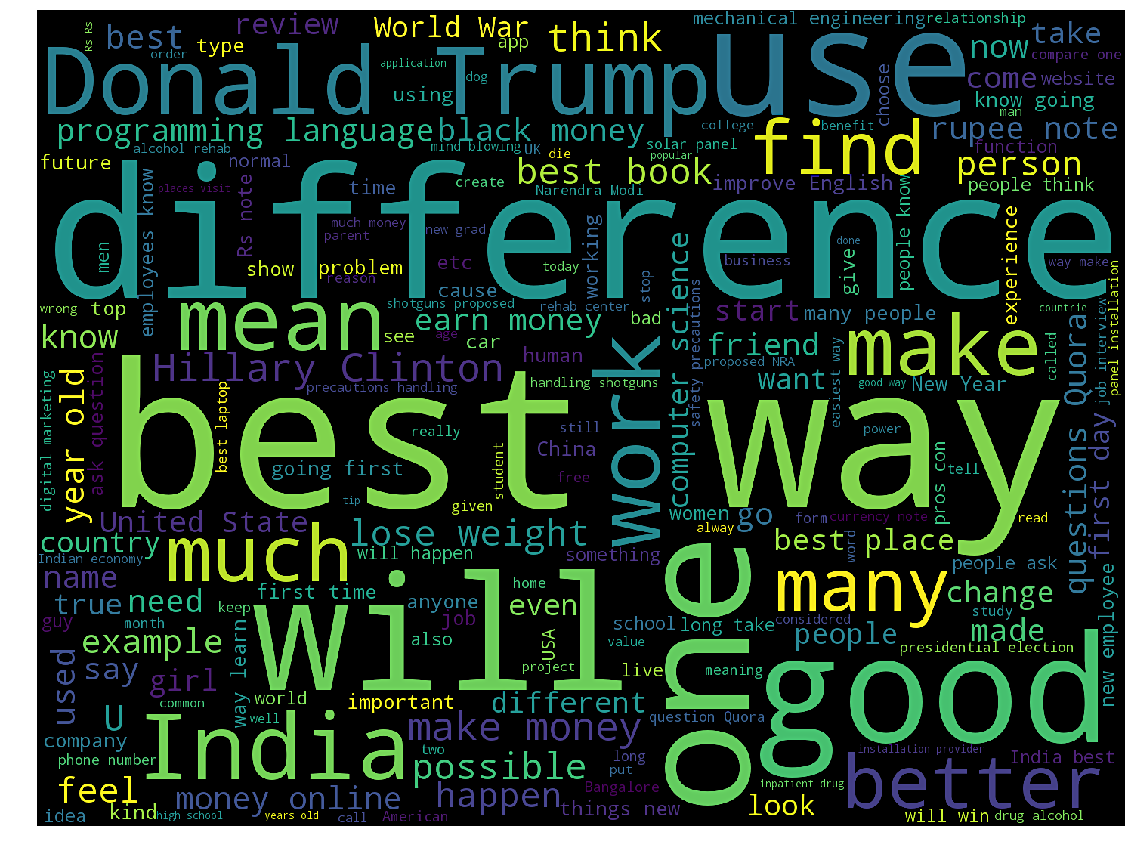
From a high level perspective, the system needs to perform the tasks in the order as shown in the architecture. The questions that need to be investigated are extracted from the dataset, text cleaning and pre-processing is performed for best results, the sentences are represented in a different for such that a similarity measure can be applied.

## *Data Set*

The data for this project has been extracted from <https://kaggle.com> [1], the home for data science. The data, as is, was provided by Quora for their Kaggle competition, Quora Question Pairs. Each instance of the dataset provided has two questions and a human annotation of whether the pair is a duplicate pair or not.

## *Text Cleaning and Pre-processing*

The dataset, when analyzed displayed high occurrences counts of prepositions, conjunctions, question words (what, how, why etc.), and some other types of high frequency words which do not change the semantic representation of the text. Hence, these *Stop Words* were removed from being processed into further representations of the text. An analysis of text after the removal of these *Stop Words* was performed and a word cloud was generated as in Fig. 2.



1. Word Cloud using high frequency words

## *Representation of sentences*

This project deals with three different techniques to represent the sentences. The first method explores the TF-IDF measure, as in [7], of representing words in a document. Sentences are represented in vector format, using the TF-IDF values of the words. The similarity measure to identify closeness among sentences was then used over those vectors.

The second method works towards developing a sentence formed by the word sense disambiguation using the adapted Lesk algorithm, as in [2], which is a slight modification from the Micheal Lesk Algorithm [3]. Some other similarity measures were explored with this representation of the questions and two of those were selected based on availability and ease of use.

The third method works on extracting the vector representation of the complete sentence as a whole. It uses the doc2vec (or paragraph vector representation of text as in [11]) for representing sentences as a complete vector instead of single words being used as elements of a vector. Section IV explains how these methods have been implemented and specific reasons of why they were chosen.

## *Similarity Measures*

This project uses the cosine similarity measure wherever the sentence is being represented in the form of vectors. with three different techniques to represent the sentences. The first method explores the TF-IDF measure, as in [7], of representing each word and how important it is for a set of documents. The cosine similarity measure is based on the cosine distance which can be represented as in Eq. (1). It is a representation of two non-zero vectors of the same shape. Given two vectors of attributes, A and B, the cosine similarity, , is represented using a dot product and magnitude as

where and are components of vectors A and B respectively.

The semantic representation of text using the Lesk algorithm calls for a similarity measure which is based on the semantic distance between the senses of the words as expressed by the word sense disambiguation of the sentences. The NLTK [4] houses implementation of some of the similarity measures which return the semantic similarity between two senses when comparing a Noun with a Noun, and a Verb with a Verb. Two of these measures, namely, Path Similarity and Wu and Palmer Similarity [12]. These measures were used as is from their implementation in NLTK while using WordNet [8].

# Implementation

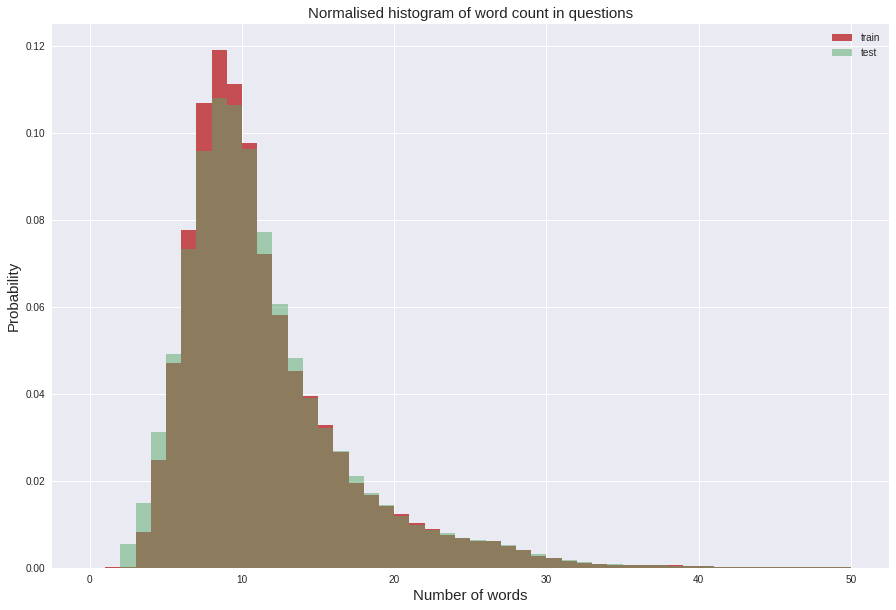
This section describes the implementation and use of each of the methods discussed in Section III and gives justification of why they were used. It also describes each of the methods in detail.

## *Data Set*

On investigation of the data, some of the following information was extracted:

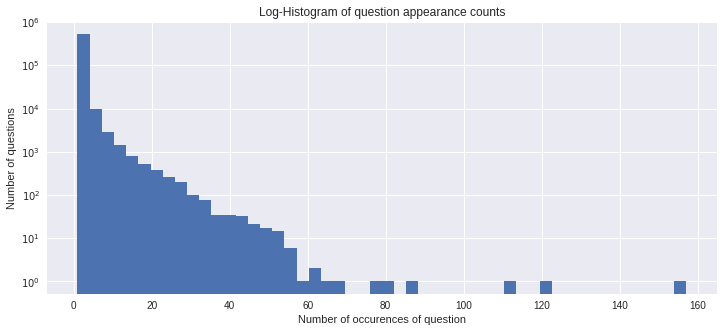
Questions with question marks: 99.87%  
Questions with [math] tags: 0.12%  
Questions with full stops: 6.31%  
Questions with capitalized first letters: 99.81%  
Questions with capital letters: 99.95%  
Questions with numbers: 11.83%

The word distribution in the dataset, including both train and has been shown in Fig. 3.

****

1. Normalized histogram of word counts in questions

There were also some questions in the dataset, which were repeated more than once. Fig. 4 gives a histogram of the occurrences of questions in the image.

****

1. Log-histogram of question appearance counts

Some of these analytics made way for an interpretation of what representation methods should be used over the questions and how to represent the pairs in order to extract the most amount of information from the sentences.

## *Cosine Similarity with TF-IDF weighting*

The first approach that was used was to represent the sentences in the form of vectors. The simplest method to convert the sentences to vectors was to assign a numerical value to each of the words in the sentence and consider the sentence as a complete vector. This method was used as it serves as a baseline for the project and any method performing worse than this should in fact be rejected.

The TF-IDF, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect the importance of a word to a document in a collection of corpus. The TF-IDF value increases proportionally to the number of times a word appears in a document, but is inversely proportional to the frequency of the word in the corpus. Mathematically,

(2)

(3)

where is the frequency of term *t* in document *d*,

is the total number of documents in the corpus, and,

is the number of documents where the term *t* appears in the corpus. can be zero, if the term is not in the corpus and might lead to a division-by-zero error, hence, the *+1* smoothing has been applied. The final TF-IDF value is represented as Eq. (4).

(4)

The tf-idf value is calculated for each word and a vector of these values is treated as the sentence vector for measuring the similarity. The cosine similarity gives a simple measure between two vectors and came in handy with these sentence vectors.

## *Semantic Similarity*

The second approach applied to represent sentences performed word sense disambiguation of the sentences and compared the senses using path similarity and Wu and Palmer similarity. This approach used the following sequence of steps:

* Tokenized the sentences.
* Part-Of-Speech tagged the sentences.
* Extracted only the Nouns, Verbs and Adjectives in the sentences (also a pre-processing step).
* Applied Word Sense Disambiguation of sentences using the Adapted Lesk Algorithm (Part D).
* Given two sentences X and Y. Build a semantic similarity relations matrix R[m, n] of each pair of word senses, where R[i, j] is the semantic similarity of the most appropriate word sense of the word at th position in X and most appropriate word sense of the word at th position in Y. The semantic similarity relations matrix is calculated using both the Path Similarity and Wu and Palmer Similarity and the average of the two similarity values is recorded in R[m, n]. If a word does not exist in the dictionary, the edit-distance similarity measure has been used to record the value.
* The match results from the previous step are combined into a single similarity value for the two sentences. There are many strategies to acquire an overall similarity of the two sets of lexicons. A computation critical algorithm, Algorithm 1, has been used to compute the final output score for the two sentences. It takes in the semantic relations matrix and the two sentences X and Y and returns a value of the similarity.



## *The Adapted Lesk Algorithm*

The Michael Lesk algorithm [3] uses dictionary definitions to disambiguate a polysemous word in a sentence context. The objective of this algorithm is to count the overlap of words in the gloss for context of the word and the definitions of the word and return the best matching sense of the word. The adapted Lesk algorithm [2] proposes some improvements over the original algorithm. This project adapts some of those proposed changes and implements another fast computing adapted Lesk algorithm following the given sequence of steps. As a pre-requisite, we will need an easily accessible dictionary of words and their definitions. In this case, the WordNet was used.

* The context of the word after the pre-processing steps is now composed only of Nouns, Verbs and Adjectives as screened after the Part-Of-Speech (POS) tagging step.
* For each word in the selected context, looked up and listed all the possible senses of both POS, noun and verb.
* Combine all possible gloss pairs that are archived in the previous step and compute overlap. The overlap is the sum of the scores of each relation pair. If the sense of some word in the context has already been calculated, then the overlap computation discards all other senses and computes overlap using only that particular best sense of the context word.
* Once each combination has been scored, we return the sense that has the highest score to be the most appropriate sense for the target word in the selected context.

## *Cosine Similarity using Doc2Vec*

Doc2Vec is an NLP tool for representing documents as a vector and is a generalizing of the Word2Vec method. The paragraph or document vectors are asked to contribute to the prediction task or the next word, given many contexts sampled from the paragraph, or in this case, question. Every paragraph is mapped to a unique vector, represented by *D* and every word is also mapped to a vector, represented by *W.* The paragraph vectors and the word vectors are concatenated or averaged to predict the next word in the context.

The advantage of the paragraph vectors is that they take into consideration the word order, at least in a small context, in the same way that an n-gram model with a large n would do, unlike the TF-IDF method which only represents a single word at a time. This comes into importance as the n-gram models preserve a lot of information of the sentence, including the word order. Fig. 5 represents the paragraph vector framework as proposed by Le, Quoc V., and Tomas Mikolov in [11].



1. The framework for representing paragraph vectors.

Another advantage of using paragraph vectors is that they can be represented in any size. A paragraph vector for a sentence may take any length we want it to represent. This was also of importance considering the fact that most of the question pairs were not of the same length and the Doc2Vec method brought them to the same length, unlike the extra computation that had to be performed in the TF-IDF vector representation. The project uses the maximum word count in a question, 50, to represent the Doc2Vec vectors.

# Experiments and Results

All of these methods were implemented and applied to the dataset. The methods as can be understood are all unsupervised methods which did not require any annotation of the data. But, since the data used was annotated, logarithmic loss evaluation metric was applied to compare the predicted values and the ground truth.

## *Logarithmic Loss for Evaluation*

Logarithmic Loss is the single class version of the log-loss metric. The metric is negative the log likelihood of the model that says each test observation is chosen independently from a distribution that places the submitted probability mass on the corresponding class, for each observation. The logarithmic loss for a single prediction with respect to a ground truth can be represented as

(5)

Logarithmic Loss for N instances can be represented as

(6)

| Model | Logarithmic Loss |
| --- | --- |
| Cosine Similarity using TF-IDF | 0.6649 |
| Semantic Model (Unscaled) | 0.6497 |
| Semantic Model (Sigmoid Output) | 0.7241 |
| Cosine Similarity using Doc2Vec | 0.6897 |

1. Log-Loss Outputs

Notice the changes in logarithmic loss value in Table I. as we switch from the TF-IDF representation of the sentences to the Doc2Vec representation of sentences. Also, the unscaled semantic model output outperforms all of the other models, in lieu of the fact that it is using specific word senses for semantic representations of sentence. This may also be because the values in the unscaled output in the semantic model are in the range [0, 0.5] and the others output in the range [0, 1]. The sigmoid version was added to output values in the range [0, 1] which then performs the worst. This might be possible as the semantic model does not take into consideration words which are not Nouns, Verbs or Adjectives, while the other models consider all the words (excluding stop words) in the questions.

## *Confusion Matrix for Evaluation*

Al of the three models were also evaluated using a second evaluation criteria in case the first was not good enough or lacked in some aspects. In the classification task of these models, it was assumed that the threshold function for classification split on the value of exactly 0.5. More investigation revealed that the accuracy for predicting duplicates also depended on the threshold value which is set for classifying as duplicate or not.

Table II. shows the normalized confusion matrix for cosine similarity using TF-IDF model. Notice how the accuracy for classification of duplicates and not duplicates changes as the threshold value increases or decreases as the threshold value is increased or decreased. Also notice a mean accuracy of around 65% for both duplicates and not duplicates when the threshold is set to 0.5.

1. Confusion Matrix (Normalized) for TF-IDF model

| Threshold Value | 0.25 | | 0.5 | | 0.75 | |
| --- | --- | --- | --- | --- | --- | --- |
| Duplicate | 0.38 | 0.62 | 0.65 | 0.35 | 0.85 | 0.15 |
| Not Duplicate | 0.08 | 0.92 | 0.33 | 0.66 | 0.65 | 0.35 |

1. Confusion Matrix (Normalized) for Semantic (Sigmoid Output) model

| Threshold Value | 0.25 | | 0.5 | | 0.75 | |
| --- | --- | --- | --- | --- | --- | --- |
| Duplicate | 0.15 | 0.85 | 0.52 | 0.48 | 0.83 | 0.17 |
| Not Duplicate | 0.05 | 0.95 | 0.24 | 0.76 | 0.65 | 0.35 |

1. Confusion Matrix (Normalized) for Doc2Vec model

| Threshold Value | 0.45 | | 0.5 | | 0.6 | |
| --- | --- | --- | --- | --- | --- | --- |
| Duplicate | 0.08 | 0.92 | 0.49 | 0.51 | 0.98 | 0.02 |
| Not Duplicate | 0.07 | 0.93 | 0.46 | 0.54 | 0.91 | 0.09 |

Table III. shows the normalized confusion matrix for the semantic model. Notice how the change in accuracy for classification of duplicates and not duplicates is similar to what can be observed in Table II.

Table IV. shows the normalized confusion matrix for cosine similarity using Doc2Vec model. Notice how quickly the accuracy for classification of duplicates and not duplicates changes as the threshold value increases or decreases as the threshold value is increased or decreased. Also notice a mean accuracy of around 50% for both duplicates and not duplicates when the threshold is set to 0.5, which means it did not make it any better. This may also be because of the use of bag-of-words model in the training of doc2vec, which does nothing different than training.

The next section gives further explanation of the results and why these do not seem to improve for some model.

# Conclusion

The project experimented with different techniques to identify question pairs with intention equivalence. All of the techniques attack a different area of machine learning or information extraction from text. Two of the methods represent convert the representation of text in the sentences to a representation of numerical values to compare them using cosine similarity. The third uses basic NLP semantic techniques to learn the senses of the two sentences and compare how close the senses are.

For all of these algorithms, a lacking factor is their inefficiency to understand the text and extract relations among various words to know which word or phrase is an important. Let’s take for example the first question pair in the data which is comparing “What is the step by step guide to invest in share market in India?” and “What is the step by step guide to invest in share market?”. As is obvious from the words in the questions, these two share about 90% similarity. The TF-IDF is supposed to capture the importance of the term “India” by applying to it a larger weight, whereas, in fact the TF-IDF model outputs a cosine similarity of 0.96 between these two questions. Similar would be the case with Doc2Vec, which, even though captures the context, is not able to identify the important term in the sentence.

Using the semantic model on these two questions uses the term “India” as a Noun after POS tagging of the words. Since India is a Proper Noun and there is no gloss of this term in the WordNet, the term is actually compared to every other term in the second question using the edit-distance similarity, which is only effective if both the terms are the same. So, this model is not even slightly able to identify the fact that the term India is the important word in the first sentence which separates these two sentences from each other.

There is so much more that can be done in this project. The data set provided has been annotated by humans and can be used to produce supervised versions of the methods implemented using machine learning algorithms. The features of these machine learning algorithms can be extracted using the methods implemented above and extensions of these methods by adding probabilistic models such as Naïve Bayes or Bayesian Networks. With this, various methods of information extraction or document summarization might be of much help in extracting the important phrases or intent for the questions or sentences.

# References

1. <https://kaggle.com> Kaggle, the home for Data Science
2. Banerjee, Satanjeev, and Ted Pedersen. "An adapted Lesk algorithm for word sense disambiguation using WordNet." *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer Berlin Heidelberg, 2002.
3. Lesk, Michael. "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone." *Proceedings of the 5th annual international conference on Systems documentation*. ACM, 1986.
4. Bird, Steven. "NLTK: the natural language toolkit." *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 2006.
5. Bhagwani, Sumit, Shrutiranjan Satapathy, and Harish Karnick. "Semantic textual similarity using maximal weighted bipartite graph matching." *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2012.
6. Dao, Thanh Ngoc, and Troy Simpson. "Measuring similarity between sentences." *WordNet. Net, Tech. Rep.* (2005).
7. Achananuparp, Palakorn, Xiaohua Hu, and Xiajiong Shen. "The evaluation of sentence similarity measures." *International Conference on Data Warehousing and Knowledge Discovery*. Springer Berlin Heidelberg, 2008.
8. Miller, George A. "WordNet: a lexical database for English." *Communications of the ACM* 38.11 (1995): 39-41.
9. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research* 12.Oct (2011): 2825-2830.
10. Řehůřek, Radim, and Petr Sojka. "Gensim—Statistical Semantics in Python." (2011).
11. Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." *ICML*. Vol. 14. 2014.
12. Wu, Zhibiao, and Martha Palmer. "Verbs semantics and lexical selection." *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994.