Duplicate Detection in Quora Question Dataset

Amir Rahimzadeh Ilkhechi, Andrew Lee, Srikar Pyda, and Usama Naseer

Duke University

May 5, 2017

Abstract

Entity resolution, also commonly referred to as deduplication or dedup, is the process of identifying duplicate entries within a dataset. Our class project focuses on identifying duplicate questions in the popular Q&A website Quora. We leverage two existing toolsets called word2vec and doc2vec to analyze question semantics and convert each word/question into a vector. In vector form we are able to operate on a word/question and compare it with other word/question vectors to extract similarities. We present five approaches to identifying duplicates in the Quora dataset. The first four make use of commonly used machine learning algorithms such as Random Forest and Neural Networks after vectorizing the words. The final strategy naively uses a threshold and labels the given question pair accordingly. While the machine learning methods performed modestly well, the naive threshold strategy outperformed better by a significant amount.

1 Introduction

Entity resolution is the process of identifying and merging records which refer to the same "real-world" entity. For example, a database may contain two records which refer to same individual, "John Locke," but spell his first names different: "John Locke" and "John Lock." There are three "big-picture" problems that entity resolution address: deduplication, clustering records which refer to the same entity, record linkage, matching records from a deduplicated dataset to another (relevant in the context of relational databases), and reference matching, matching noisy record to clean ones in a deduplicated reference table. Entity resolution aims to reduce redundancies within databases; grouping duplicates facilitates efficient knowledge sharing. With the recent wave of research in Big Data and Machine Learning, entity resolution is especially pertinent for companies to minimize redundant records, data inconsistencies, and accuracy of statistical analysis over the dataset. Entity resolution often reduces the complexity across networks and semantic relationships between entities, which promotes efficiency as the database continue to grow. Enterprise companies thrive on the accuracy and consistency of the data they provide to their customers; consumers prefer accessibility and reliability over the technological nuances of their product. Entity resolutions frameworks tend to be oriented towards the database it is cleaning; the challenge of deduplication is that the errors are heavily dependent on the format and semantic content of the database and the entries it contains.

Quora, Inc., was founded in 2009 by Adam D'Angelo and Charlie Cheever in Mountain View, California; it hosts Quora, a question-and-answer site where questions are asked and answered by its users. Quora's mission is to "Quora's mission is to share and grow the world's knowledge... Quora has only one version of each question... Quora's answers come from people who really understand the issues and have first-hand knowledge.[7]" Quora distinguishes itself from competitor information forum websites by focusing on the quality of information provided: they require users to register with their real names as opposed to pseudonyms and prevent anonymous up-votes, aiming to increase answer credibility[6]. As a result, entity resolution is particularly pertinent for Quora because it directly relates to their mission of providing users with accessible, reliable knowledge.

We aim to develop an entity resolution framework for the Quora dataset of questions and answers: our project focuses on identifying duplicate questions within the Quora database. We developed a multitude of algorithms which aimed at identifying questions with differing syntactical form but similar semantic content: a term-frequency-inverse document frequency (TF-IDF) pair-wise matching approach, a TF-IDF pair-wise matching approach with clustering, an intersection-based approach, and finally, an approach revolving around Doc2Vec [2]. While extracting sentence similarity-features based on the previous methodologies, we experimented with a multitude of vector similarity metrics: Cosine Similarity, Jaccard's Similarity, Euclidean Distance, and Manhattan Distance. Finally, we trained three primary Machine Learning models on the features extracted: Neural Networks, Random-Forest, and K-nearest neighbors. We evaluated our results based on their precision and accuracy. Overall, our final approach premised on Doc2Vec was most effective, with a precision rate of .54 and a recall rate of .87.

2 Motivation

Quora prides itself on the quality of information it provides users; resolving duplicate questions is one of the major challenges the company is facing as they are growing scalability concerns become more pertinent. Duplicate questions deter users from posting in-depth responses, particularly for challenging and technical questions, because of the nuisance that they will encounter a similarly worded query later. Entity resolution motivates users by guaranteeing that their answers will definitively provide information for anyone browsing the site with a relevant query. Limiting the number of duplicate questions maximizes the chances that users are provided with the full-spectrum of answers to their query. However, without entity resolution, users may have to traverse through a multitude of similarity worded questions to find the information they are looking for. Furthermore, redundancies diminish the utility of Quora's upvote indicator for reliability of information because they will be dispersed between posts. Deduplication will facilitate increase user satisfaction through promoting easier accessibility and improving the quality of their feedback process.

On January 24th, 2017, Quora released their first public dataset of over 400,000 lines of potential question duplicate pairs based on actual questions from Quora; each record contains: IDs for each question, the full text of each question, and a Boolean value indicating whether the two questions constitute a duplicate.[4] Although they do not guarantee perfect ground truth, Quora released this dataset so that researchers could develop an entity resolution framework revolving around issues of natural language programming, machine learning, and scalability: "An important product principle for Quora is that there should be a single question page for each logically distinct question.[6]"

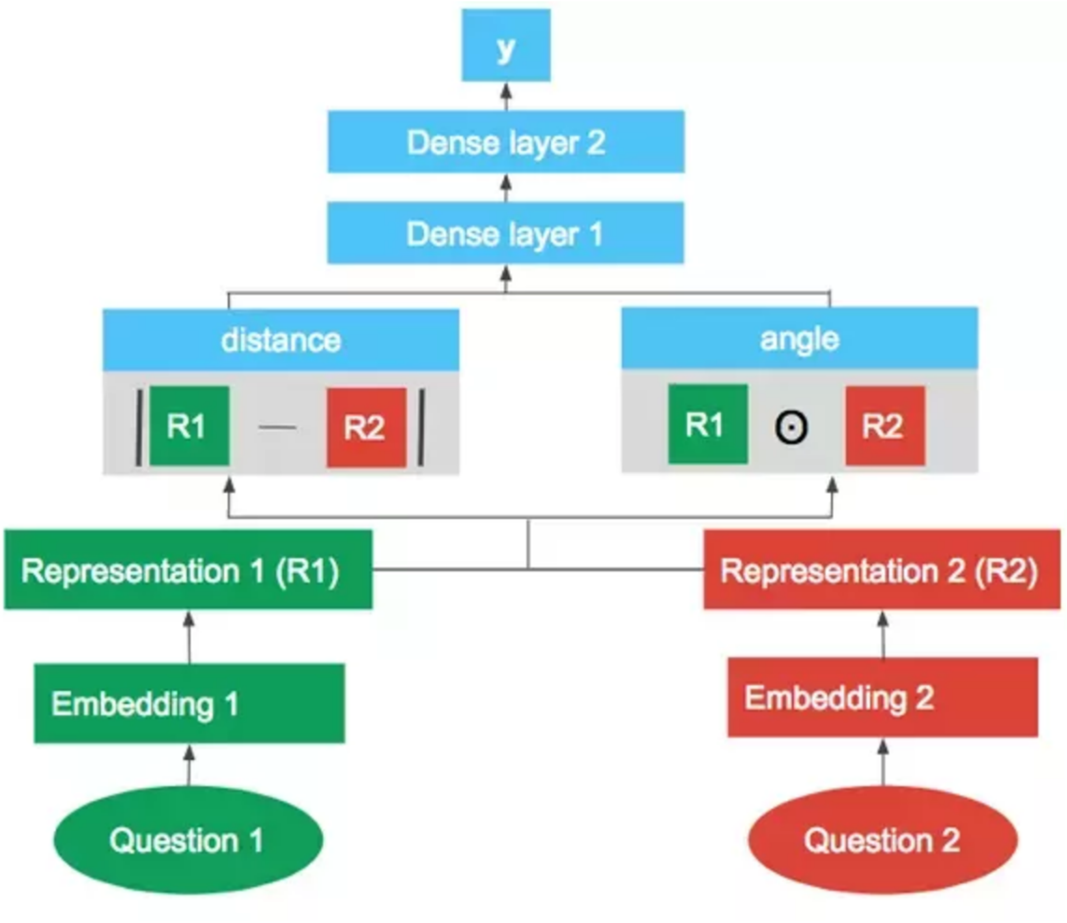
We are interested in solving this manifestation of entity resolution because it requires an analysis of character data with semantic content that is normally applied to numerical analysis. Although not completely cleaned, we found Quora's public dataset to provide the most relevant information for feature extraction in the simplest, most consistent schema. Furthermore, this variant of entity resolution touches on a multitude of topics covered in class: deduplication, clustering, scalability, and query-driven approaches. The four of us come a variety of backgrounds: Big Data, Machine Learning, Computer Networks, and Distributed Systems. As a result, this question intrigued us because it allowed us to synthesize a multitude of algorithms which aimed at isolating the semantic similarities and differences between questions which differentiate between "duplicate" and "non-duplicate" categorization. We hope that the results from our project will contribute

3 Related Work

3.1 Overview

Our project is at the intersection of both entity resolution is a well studied area, and natural language processing, both of which have been well studied in their respective areas. Natural Language Processing (NLP) focuses on the interaction between human language and computers; NLP aims at analyzing, understanding, and determining the semantic content of computational representations of human language. There is a multitude of research currently being done to determine the semantic content and similarity of sentences. These approaches often rely on providing a set of features to train Machine-Learning models for classification; we focus on exploring Neural Network and Random Forest models. Neural Networks are a biologically inspired paradigm of machine learning which is based on synapses taking an input, multiplying it by a weight, and calibrating until the weights can accurately predict an output. Deep learning is a powerful set of techniques for learning in neural network; doc2vec and doc2word are common tools which provide vectorized representations of sentences and individual words. Random Forests operate through constructing a multitude of decision trees at training and outputs the class which the mode of the classes. A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision. All of the work in this project builds off recent advances in machine learning and NLP.

Figure 1: Figure 1: Architecture of approach 1, “LSTM with concatenation”[5]



3.2 Quora’s Initial Approach

3.2.1 Overview

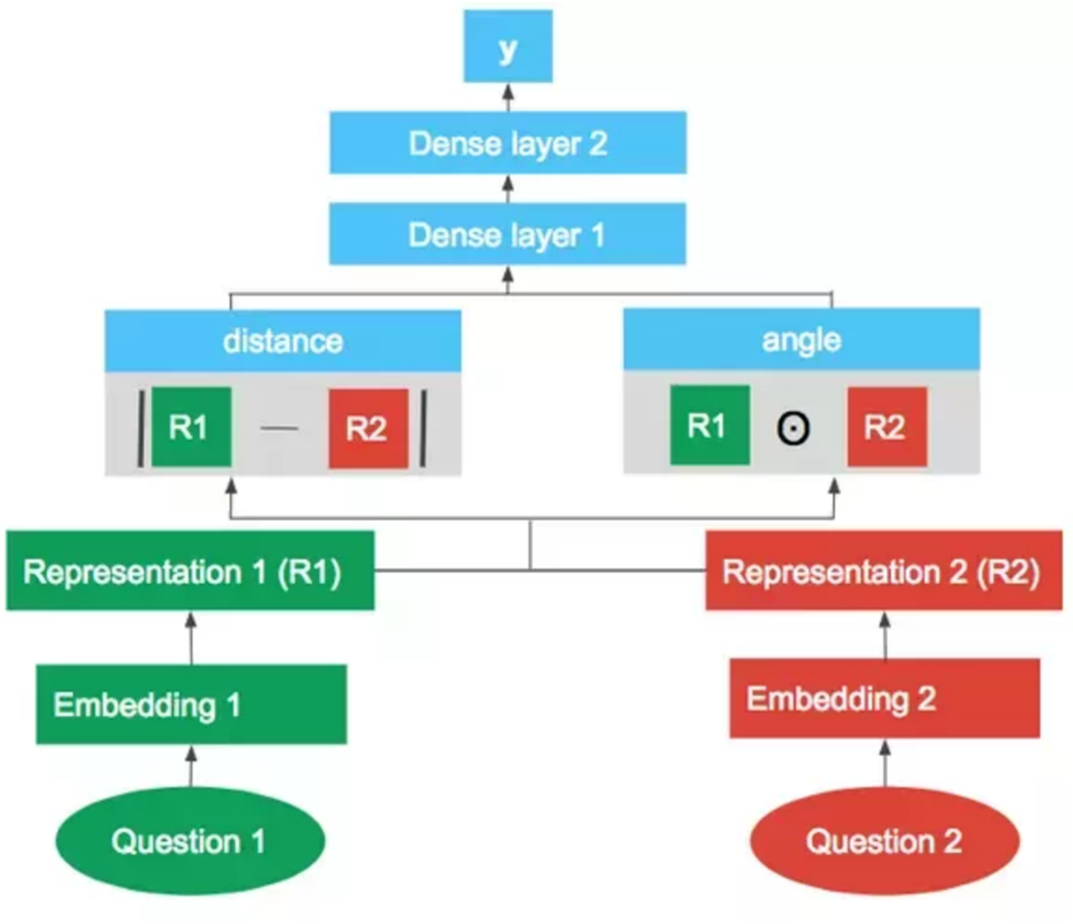
After releasing the public dataset, Quora provided three initial approaches based on a random-forest model with features including cosine similarity of the average of the word2vec embeddings of tokens, the number of common words, the number of common topics, and the part-of-speech tags of words within the sentences. Quora’s proposed algorithms focus on both scalability and accuracy by avoiding iterative and computationally complex feature engineering through an end-to-end deep learning solution. They formally define the question: given a pair of questions, train a model which identifies them as either duplicate or non-duplicate.[5]

The simplest approaches to the comparison of similarity between strings comes from common word-based comparison metrics such as tf-idf and BM25; these approaches find the word-based similarity between two questions and classify pairs based on whether their similarity scores meet a threshold of semantic similarity.[5] However, these approaches do not effectively determine the intentional semantic content of the sentence as opposed to a comparison of the overall frequency of words within the sentences.

3.2.2 Approach 1

Quora’s first approach utilized the Long Short Term Memory network variant of Recurrent Neural Networks (RNNs), which captures long-term dependencies.[5] First, they trained their word embeddings using Quora’s text corpus.[5] Then, they combined the word embeddings to generate the question embeddings for the two questions.[5] Afterwards, they fed those question embeddings into a representation layer.[5] Finally, they concatenated the two vector representations of the questions and feed it into a dense layer which produces the final classification of the question.[5]

Figure 2: Figure 2: Architecture of approach 2, "LSTM with distance and angle"[5]



3.2.3 Approach 2

Quora's second approach implemented Thai, Socher, and Manning's choice of feature extraction: the distance, the sum of squares of the difference of the two vectors, and the angle, the element-wise multiplication of two vector representations.[5]

3.2.4 Approach 3

Quora's final approach modeled Google Research's attention-based approach which combined neural network attention with token alignment; they distinguish this approach because of the minimal number of parameters.[5] Like their other two approaches, this one represents each token with a word embedding. They utilize Google Research's "Decomposable attention" model to then train a soft alignment attention model for all pairs of words, compare aligned phrases, and aggregate comparisons for classification.[5]

4 Research Question

4.1 Overview

We looked at a variety of datasets to use for deduplication, including StackOverflow's and Quora's questions and answers corpus. We required the dataset to be labeled for training and text based. After experimenting on various datasets, we decided to go ahead with Quora's questions corpus for the purpose of testing our dedupication techniques. Quora's dataset is text-based (does not include multimedia). Although there are a few appearance of words from foreign laguages like Mandarin, Hindi, German, French etc, the corpus is mostly English based. The datset encompasses a broad range of concepts, from small trivia questions like *'What is the funniest joke of all time?'* to comparatively longer, technical questions like *'How Google helps in spam ranking adjustment of the search results?'*.

Figure 3: Figure 3: Overview of approach 3, “Decomposable attention”[5]

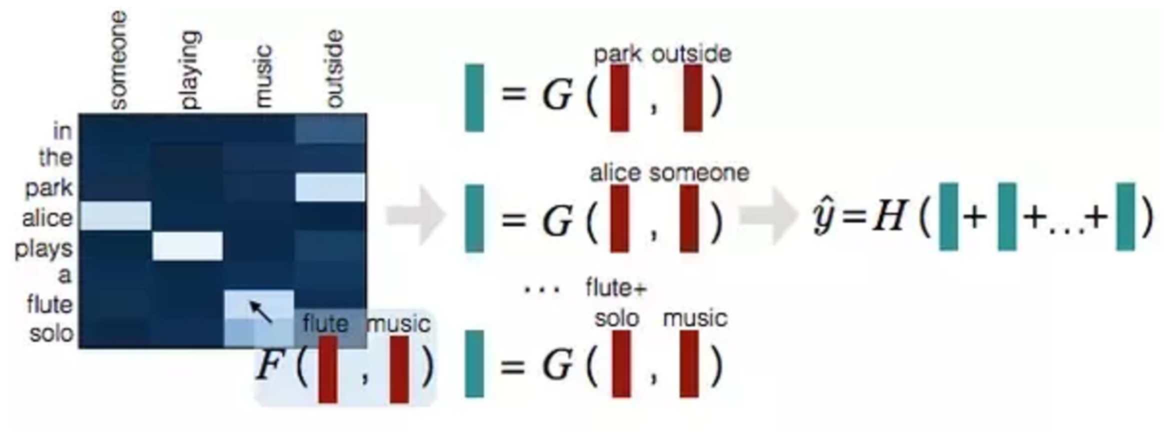
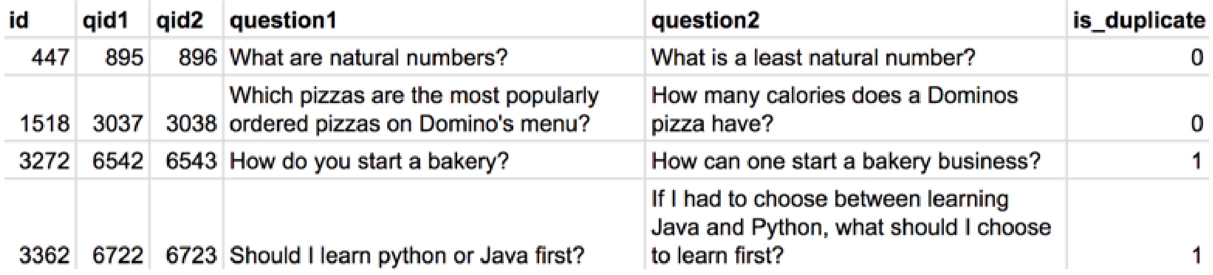


Figure 4: Quora Dataset Schema [4]



id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

4.2 "Dirty" Dataset: Quora

4.2.1 Dataset Schema:

The input data from Quora contains pairs of questions, IDs and a boolean flag. The flag identifies whether the two questions are duplicates or not. The IDs are the unique identifiers for questions.

4.2.2 Dataset Properties:

Quora’s released dataset contains over 400,000 lines of potential duplicate question pairs. We found that 20 percent of question pairs were labeled duplicates. Although most of the questions were in English, there were some other languages.

Their original sampling method returned an imbalanced dataset: there was significantly more true examples of duplicates than non-duplicates.[4] As a result, they supplemented the dataset with examples of non-duplicate pairs; for example, questions relating to similar topics with differing semantic content and combined a multitude of sampling techniques.[4] Furthermore, they applied a variety of pre-processing methods to sanitize the data. Therefore, the distribution of questions in the published dataset do not necessarily represent the overall distribution of questions on Quora.

Quora notes that the ground-truth labels are not perfect.[4] While manually examining the dataset, we observed many instances in which question pairs were falsely labeled as duplicates. This impacts the accuracy of our machine learning models because they assume absolute ground truth.

4.2.3 Data preprocessing:

In preprocessing, we cleaned the dataset to remove meaningless symbols like semicolons, exclamation marks etc, since these symbols can degrade similarity between two words. Eg, *happy* and *happy!* convey the same meaning but the attached-symbol makes them different when similarity metrics like cosine similarity is calculated between the two. Quora’s dataset does not include HTML, which makes the HTML tag stripping redundant. However, it was important that we distinguished non-alphanumeric characters which add semantic value to a sentence and did not sanitize them. For example, quotation marks of indicate referential and conversational content while parentheses indicate explanatory material.

Metadata:

Metadata for a dataset includes information about the user posting the questions, date, answers etc. The original Quora dataset does not include any kind of metadata. We did considered web scraping to get question's metadata but ultimately decided to not include metadata in our deduplication model. The reason of excluding metadata is that the answers might not always be closely related to the questions.

4.3 Formal Statement of Problem

Given a pair of questions *q1* and *q2*, train a model that learns the following function:

$$f(q_1, q_2) -> 0, 1$$

where 1 represents that *q1* and *q2* are a pair of duplicate questions and 0 represents otherwise.

5 Approach

5.1 Overview

We implemented four distinct algorithms aimed at identifying duplicate question pairs based on their semantic similarity. Most of our approaches trained a Machine Learning model based on the extracting of a set of features for each question pair. We had to provide vectorized representations of strings so that we could perform similarity comparisons. As a result, our approaches relied heavily on the accuracy of deep-learning tools such as word2vec to vectorize both individual words and whole sentences. Our first three approaches differ based on the features extracted to determine semantic similarity. Our final approach does not follow this model and is similar to a word similarity comparison outlined as a "basic" method above.

5.2 Tools

As discussed above, our approaches build upon a multitude of existing deep-learning tools which facilitate the vectorization of words and entire documents.

Word2vec [1] is a group of two-layer neural networks that are trained to reconstruct linguistic contexts of texts. Word2vec takes a large corpus of text as an input and outputs a multi-dimensional vector space, which each word being assigned a corresponding vector. Vectors are positioned such that words which share similar contexts within the corpus are located in close-proximity. This tool allowed us to produce a 100-dimensional vector for each word within the corpus-text.

Deeplearning4j (DL4J) [3] is an open-source, distributed deep-learning project in Java and Scala developed by Skymind, a San Francisco-based software team. We utilized their library to develop a document vectorizer which produces a 100-dimensional vector for each question.

5.2.1 Similarity Comparison

Similarity is a measure of how alike two data-object are; it is measures in the range between 0 and 1, where 1 represents that the two objects are identical while 0 represents that the twoo objects have no similarity. When comparing vectors, whether they represent individual words or whole sentences, we utilized a multitude of similarity comparison metrics: Cosine Similarity, Jaccard's Similarity, Euclidean Distance, Manhattan's Distance, and Minkowski's Distance.

Cosine Similarity finds the normalized dot product of the two attributes. It effectively finds the cosine of the angle between the two vectors. A cosine value of $0°$ indicates a similarity value of 1. This metric measures similarity based on orientation as opposed to magnitude. Independent of magnitude, two vectors with identical orientation have a cosine similarity of 1, two vectors at $90°$ have a similarity of 0, and two vectors diametrically opposed have a similarity of -1.

Jaccard's Similarity measures the similarity between finite sets by the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets.

Euclidean Distance is the most common use of distance. The Euclidean Distance between two points is the length of the path connecting them.

Manhattan Distance is a metric in which the distance between two points is measured by the sum of the absolute differences of their Cartesian coordinates; it is the total sum of the difference between the x and y coordinates.

Minkowski's Distance is a generalized metric form of Euclidean distance and Manhattan distance.

5.3 Approaches

We first introduce some terminology that we will refer back to later in the section to explain the feature set generation process. Let $Q_1 = \{w_1, w_2, \dots w_n\}$ be a set of n words, and $Q_2\{w_1, w_2, \dots w_m\}$ be another set with m words. Assume that every word is lowercase and any punctuation has been removed. Using the word2vec tool, each word w_i can be represented as a 100-dimensional vector $w_i = (v_1, v_2, \dots, v_{100})$. We will use the vector representation of each word for all future discussion and analysis.

5.3.1 Naïve Strategy

In order to establish a baseline using word2vec, we employ a naive strategy for detecting duplicates. Given the vector representation of every word in a question, we add up the vectors and compute the cosine similarity between these two questions. We construct a single new vector

$$w^* = w_1 + w_2 + \cdots + w_n = (v_{w_1,1} + \cdots + v_{w_n,1}, \dots, v_{w_1,100} + \cdots + v_{w_n,100})$$

5.3.2 Approach 1: TF-IDF Pair-based Comparisons

Overview

The first strategy we used was based on generation of keywords using TF-IDF. We build dictionary of all the words present in our cleaned corpus, keeping the count of their appearance in the corpus. The lesser the number of times a word appears, the more important is the word. By 'importance' we mean that since the word appears in lower number of questions, its probability to be decisive should be greater than a more common word. Using TF/IDF we are able to remove words like 'the', 'for', 'are' etc, due to their higher number of appearances in the dictionary.

Methodology

Once we get the keywords for a question, we select the top 10 keywords (10 words appearing the least number of times in the dictionary) for comparisons. A simple, string matching does not work since it does not capture the meaning of word. For example, hatchback and coupe are closely related words. A string based matching of both will either result in 1 (exact match) or 0 (no similarity), since it does not take into account the word concept. Word2vec comes into play at this moment to solve this issue. Similar words will have similar vectors and similarity measurement (eg, cosine similarity, Jaccard's similarity) of these vectors will have a value closer to 1.

Once we have vectors representing the 10 keywords from each question, we do 10 X 10 cosine similarity measurement for the two questions by computing the cosine similarity between each pair of word vector across the two sets. As a result, we get a 100 cosine similarities for a pair of question. Using the given duplicate flag from the input dataset, we feed these 100 values and the boolean duplicate flag to our machine learning model. Till now we have tested Random Forests, Support Vector Machines and Neural Networks as learning models.

Feature Extraction

Similarity Score

We experimented with using all five of the similarity metrics described above while extracting features to calculate similarity scores.

We experimented with aggregating the above results in order to limit the number of features; we thought that decreasing the number of features will limit the amount of "noise" for the ML model. We aimed to represent only the most significant similarity comparisons as opposed to feeding the model every single pair-wise computation.

Full Feature Extraction We extracted 101 features for each question-pair; 100 features for each pairwise comparison between the two sets of 10 key-words and a final feature indicating the ground-truth value for whether the pair is a duplicate.

Top-3 Feature Extraction We extracted 4 features for each question-pair; the top three similarity scores out of all the pairwise comparisons and the ground-truth value indicating whether the question pair is a duplicate.

5.3.3 Approach 2: TF-IDF Pair-based comparisons with clustering

Overview

In this approach, we trained a Word2Vec model with sentences in the dataset and also random English lines. We also clustered all of the words in the corpus; ideally, words belonging to each group are conceptually similar; therefore, we aim to compare words belonging to the same clusters separately in a question pair.

Methodology

Naively comparing the 10 keywords with each other leads to a total of 100 matches per question pair, which is problematic at scale. In order to reduce the number of comparisons and also make the comparisons more meaningful by comparing only relevant words in two given questions, we first applied k-means clustering to the key words to group them. We used the word2vec vectors for defining the words for k-means clustering. Figure 5 represents the intuition behind our approach #2. Two words appearing in the same cluster are more or less related to each other assuming that the trained vectors are representative of the word meanings. This reduces the number of comparisons we need to make, as we can use the clusters for comparing only those words that are related to each other. We compared the top three words in each cluster (we padded the attributes related to cluster x with 0 vectors if the corresponding question did not have a minimum of three words related to cluster x). We also experimented with aggregating the results from the pairwise comparison by taking the max or average.

Algorithm 1 Feature selection for approach #2.

```
1: keywords[] ← 10 vectorized keywords of question 1 and 2
2: C[] ← clusters
3: vectors1[] ← 3 empty vectors for each C[i]
4: vectors2[] ← 3 empty vectors for each C[i]
5: features[] ← the empty set of 90 features
6: for any i in 1:10 do
7:     c ← keyword[i].getCluster
8:     if less than 3 keywords are chosen for cluster c then
9:         add keyword[i] to group c features
10:    end if
11: end for
12: for any j in 11:20 do
13:    c ← keyword[j].getCluster
14:    if less than 3 keywords are chosen for cluster c then
15:        add keyword[i] to group c features
16:    end if
17: end for
18: for any c in C do
19:    compute pairwise similarities between all vectors in vectors1 and vectors2 that belong to c and
       save the results in features
20: end for
```

Feature Extraction

Similarity Score

We experimented with using all five of the similarity metrics described above while extracting features to calculate similarity scores.

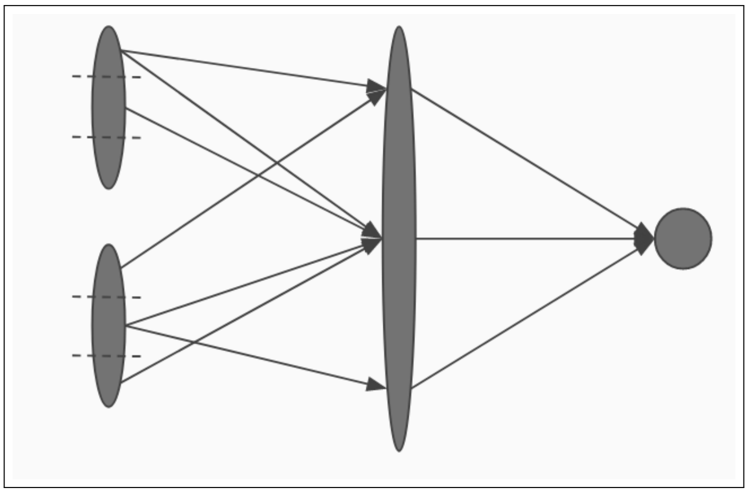


Figure 5: The intuition behind our feature selection for approach #2. Given a neural network, we can partition the inputs into two groups each representing one question. These groups are then identically grouped into feature groups. Each feature group represents the words in a question that are related to similar concepts (e.g., time, location, ...) By feeding these inputs and corresponding labels, the neural network will learn which concepts are more important when comparing a pair of questions.

We extracted all of the pair-wise computations of similarity scores as features for each question-pair in addition to the ground-truth boolean value.

Implementation

We experimented with aggregating the above results in order to limit the number of features; we thought that decreasing the number of features will limit the amount of "noise" for the ML model. We aimed to represent only the most significant similarity comparisons as opposed to feeding the model every single pair-wise computation.

Full Feature Extraction We extracted 90 features for each question-pair (we have 10 clusters, each question's top 3 words belonging to each cluster i are selected which means 3×3 comparisons for each cluster, therefore $3 \times 3 \times 10$ features in total); 90 features for each pair-wise comparison between the two sets of 10 key-words and a label indicating the ground-truth value for whether the pair is a duplicate. There are 3x3 pair-wise comparisons for the top three words in each of the ten clusters.

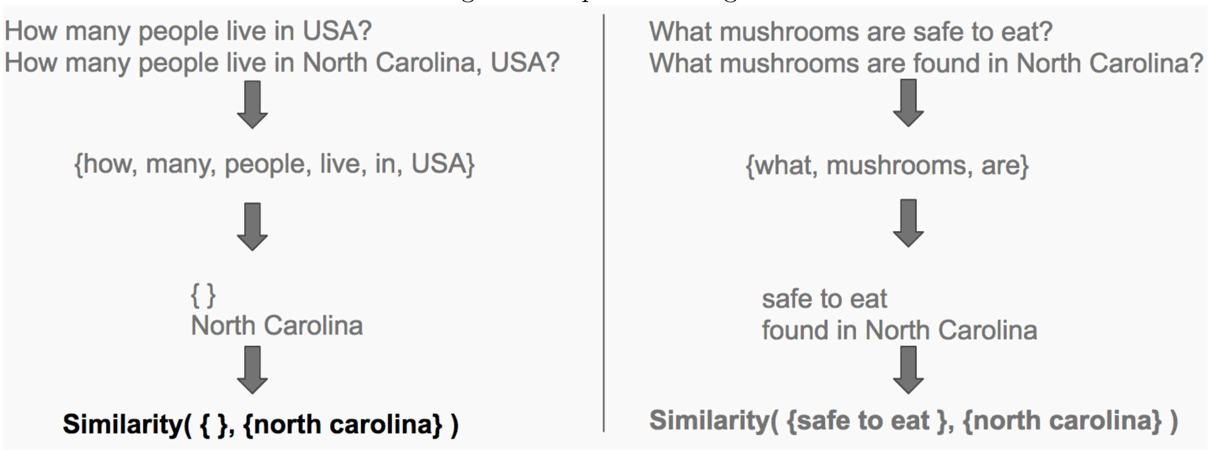
Average Feature Extraction We extracted 11 features for each question-pair, one feature for each cluster and a final feature indicating the ground-truth value for whether the pair is a duplicate. We assigned each cluster one feature, the aggregate of all pairwise comparisons within the top-3 words for each cluster.

Top-3 Feature Extraction We extracted 31 features for each question-pair; three features for each cluster and a final feature indicating the ground-truth value for whether the pair is a duplicate. The three features per cluster are extracted based on their similarity score; we select the top 3 features for each cluster.

Average of Top-3 Feature Extraction We extracted 11 features for each question-pair, one feature for each cluster and a final feature indicating the ground-truth value for whether the pair is a duplicate. We assigned each cluster one feature, the average of the top-3 similarity scores between all pairwise comparisons.

Max Feature Extraction We extracted 11 features for each question-pair; one feature for each cluster and a final feature indicating the ground-truth value for whether the pair is a duplicate. We assigned each cluster one feature, the maximum similarity score out of all the pairwise comparisons.

Figure 6: A picture of a gull.


How many people live in USA?
How many people live in North Carolina, USA?
{how, many, people, live, in, USA}
{ }
North Carolina
Similarity({ }, {north carolina})
What mushrooms are safe to eat?
What mushrooms are found in North Carolina?
{what, mushrooms, are}
safe to eat
found in North Carolina
Similarity({safe to eat }, {north carolina})

5.3.4 Approach 3: Intersection

Overview

This strategy narrowed the set of keywords considered from each pair to words that were unique to each question. We calculated the intersection between two sets of words, each containing all of the terms in a question. We defined the set of keywords of a question as the set of all words in the sentence as the difference between the original set of words in a question and the new intersection set calculates.

Motivation

We were inspired to implement this approach after analyzing Jaccard's Similarity measure, which divides the cardinality of the intersection between two sets divided by the cardinality of the union of the sample sets. We brainstormed that the semantic similarity between two questions could be measured (not perfectly) through comparing all of the terms which were unique to either question. We hypothesized that the intersection set, the set of words shared by both questions, ought not contribute significantly to the semantic content which differentiates the sentences.

We averaged the vectors of the remaining words because similar words ought to be in close proximity within the generated vector-space. As a result, a similarity comparison between the average vectors ought to provide relevant information on the semantic indicator of unique words.

Methodology

First, two "question" sets containing all of the words in each question are initialized. Then, the intersection set between the two original sets is calculated. Afterwards, intersection set is removed from both of the original "question" sets. Finally, the vectors of the remaining words in each question are be averaged together, and the resulting two aggregate vectors would be compared.

For instance consider two questions, `Who is the president of Pakistan?` and `Who is the commander of the army in Pakistan?`. The intersection of both questions is {who, is, the, of, pakistan}. Once the overlapping words are removed, we are left with `president` and `commander army`. Once the words in question two are averaged, both questions can be compared and represented by a similarity measure such as cosine similarity.

The intuition follows that the words which overlapped with each other in previous strategies provide no new information to the model. Leaving only the words that differ, we are able to isolate the only words that could change the meaning. However, unlike previous strategies, the number of words remaining could be any number and not necessarily equivalent. In the example above, Question 1 had 1 word and Question 2 had 2 words. Since similar words have similar vectors, we average the words in each question so that we can have one feature after computing the similarity.

Feature Extraction

We extracted a multitude of features for each question pair: a cosine-similarity score between the aggregate vectors for the original question-sets, a cosine-similarity score between the aggregate vectors for the modified sets after removing the intersection set from the original question sets, a Jaccard's similarity score between the aggregate vectors for the modified sets, and finally the ground-truth label indicating whether the pair is a duplicate or not.

5.4 Approach 4: Doc2Vec

Overview

Our final approach went against the intuition in the previous three algorithms. Instead of using word2vec to produce a vector space for all words in a corpus, we utilized a document vectorizes to convert each a question into a multi-dimensional vector. Rather than extracting a feature-set from similarity comparisons between sets of word-vectors to train a model, this algorithm determines classification based on whether a similarity measure between the two questions meets a certain threshold.

Motivation

We were inspired to follow this approach after realizing that we could utilize the DL4J library to produce vectors for sentences as opposed to words. We hoped that this would better contextualize sentence semantics better because it produces a vector for the sentence as opposed to individual vectors from all of the words. Sentences are more than just the sum of their words; two sentences could contain the same words but indicate differing semantic content because of grammatical structure. After experimenting, we were surprised to find high levels of precision and accuracy.

Methodology

First, each question would be vectorized one by one (we utilized the DL4J library for our implementation of the document vectorizer.) Then, the similarity between each pair of vectors was compared based on the cosine similarity measure. Finally, question pairs with a similarity score above a certain threshold are set as duplicates.

Feature Extraction

The only feature extracted was the similarity score between the two question-vectors. Unlike the other approaches, this is naive and does not include an label for categorization.

6 Results

We define three measures of performance for our models: precision, recall, and accuracy.

Precision (True Positive): The fraction of question pairs which the model identifies as a "true" duplicate out of all the entries with a ground truth value of "1."

Recall (False Positive): The fraction of question pairs which the model indicates as a duplicate that corresponds to a ground truth value of "1." out of all the entries it marked as duplicate.

Accuracy: The percentage of entries which the model accurately classifies as either duplicate/non-duplicate out of all question-pair classifications

6.1 Approach 1: TF-IDF Naive

Naive: **Random Forests**

Precision	**46.36%**
Recall	**65.48%**
Accuracy	**71.50%**

6.2 Approach 2: TF-IDF with Clustering

Clustering: **Random Forest**

Precision	**55.46%**
Recall	**73.46%**
Accuracy	**78.9%**

Clustering: **Support Vector Machine (SVM)**

Precision	**55.46%**
Recall	**73.46%**
Accuracy	**78.9%**

6.3 Approach 3: Intersection

Intersection: **Random Forest**

Precision	**0.51**
Recall	**0.39**
Accuracy	**0.72**

Intersection: **Neural Networks**

Precision	**0.50**
Recall	**0.39**
Accuracy	**0.66**

Intersection: **K-Nearest Neighbors**

Precision	**0.54**
Recall	**0.28**

6.4 Approach 4: Doc2Vec

Doc2Vec: **Threshold Similarity**

Precision	**0.54**
Recall	**0.87**
Accuracy	**0.40**

Doc2Vec: **Random Forest**

Precision	**0.74**
Recall	**0.36**

Doc2Vec: **K-Nearest Neighbors**

Precision	**0.52**
Recall	**0.79**

7 Discussion

7.1 Analysis

7.1.1 Overview

In our analysis of our results, we prioritized precision rates because the research question is concerned with the identification of "true" duplicate questions based on their semantic content. Therefore, we determine that our fourth approach, Doc2Vec, had the best performance with precision, recall, and accuracy levels of .54, .87, and .40.

Our project aimed at performing operations traditionally applied to numerical analysis to character data to determine semantic content. As a result, we discuss the individual components of our project that we believe contribute towards performing textual analysis for semantic content. Although we may not have figured the optimal configuration, we believe that our results have meaningful contributions to textual de-duplication.

7.1.2 TF-IDF

Our first two approaches utilized the TF-IDF metric for determining the keywords in each question. The TF-IDF score is inversely proportional with the occurrence of a word: words with lower occurrences are considered "unique" terms. Our goal was to isolate the words in each sentence that unique contribute to its semantic meaning; we wanted to eliminate words that were common to all sentences, for example, "but, the, a." Rather than having to do a pairwise comparison between all words, we aimed to promote scalability while preserving semantic content in adopting this procedure.

Although our first two approaches did not perform as well as the Doc2Vec approach, I believe our results still demonstrate the relevancy of TF-IDF in feature extraction. We were able to effectively select the top ten keywords from each question, reducing the number of pair-wise comparisons. Although it is unclear from the results whether semantic content was preserved, manual examination of our processing seemed to indicate that we successfully identified the top ten key-words in the question. This is particularly relevant for larger textual data which needs to be converted to a multi-dimensional vector. TF-IDF may improve the scalability of our Doc2Vec algorithm by vectorizing a sentence based on a fraction of its terms.

However, our current implementation of TF-IDF sorting is not scalable:

First, questions have to be sorted lexographically according to the most important key words

Second, sorted list has to be divided into non-overlapping canopies (worked poorly in our current implementation)

Third, the sorted list has to be divided into overlapping sets; this approach proves difficult because it is hard to find the optimal division, especially as the dataset grows

Our current implementation of TF-IDF sorting is computationally intensive and requires a far greater number of pairwise comparisons than either the third or fourth approach. TF-IDF has historically been a powerful tool in textual analysis. We believe that our results further strengthen its use as a metric to determining the "unique-ness" of a word within a corpus. Our results confirm the robust nature of TF-IDF as a tool in textual analysis: our first two approaches have significantly better recall rates than our third approach. However, all three of them perform similarly in terms of precision. This means that the first three models have similar performance in identifying "true positives." However, the third approach has a higher number of "false positives," corresponding to lower recall rates. I believe this demonstrates the importance of TF-IDF in reducing the number of words considered for pair-wise comparison. The third approach was able to detect around the same number of "true positives" as the other two approaches but may have identified more false positives because it considered more words for each question during feature extraction. However, including words with a high TF-IDF value introduces more noise to the feature-set because all comparisons involving them tend to be null. Regardless if it provides complete semantic meaning, utilizing TF-IDF in feature extraction helps improve models in detecting duplicate questions, especially by improving recall.

7.1.3 Clustering

We adopting a clustering methodology in our second approach: while training the Word2Vec model and generating the vector space, we clustered all of the vectors into 10 groups. Ideally, clusters contain conceptually similar words and words from different clusters are conceptually distinct. We hoped that clustering the key-words would reduce the number of unnecessary pair-wise comparisons. Rather than doing a 10x10 pairwise comparison between all of the keywords, we were able to do 3x3 comparisons over all 10 clusters. Although our results from the second approach were worse than the fourth approach, cluster proved to be a useful tool: not only did it reduce run-time but after manual examination it effectively clustered all of the words into 10 categories. Even though the clustering was not perfect, there was a overall trend of the semantic content of words within a cluster. We hope to improve our clustering in order to better extract semantic features from the dataset.

Clustering is a useful tool for scalability in textual analysis: rather than artificially constraining pair-wise comparisons, it only reduces unnecessary comparisons because comparing conceptually distinct terms generally does not provide insight into the semantic meaning of the sentence. Unlike numerical analysis, which can easily be statistically analyzed through a variety of aggregation methods, textual analysis requires significantly more pairwise comparisons to determine semantic content. Furthermore, textual data comparison is far more computationally complex than numerical analysis, which is built into the hardware of the machine. As a result, clustering in textual analysis significantly enhances performance by reducing the number of pairwise comparisons.

In addition to reducing the number of unnecessary pairwise comparisons, we believe that clustering the words is effective tool in gaining insight to the semantic content of textual data. Our results confirm ML models (Random Forest and SVM) from our second approach (TF-IDF with clustering) performed significantly better than our original naive approach: precision, recall, and accuracy were all higher by around ten percent. In addition to promoting an efficient extraction of features, we believe that providing features as pairwise comparisons corresponding to a cluster will better train the model. Clustering words models a human's interpretation of language; words are broken up into two types of categories: parts of speech and semantic content. Although we only address the latter in our current implementation, I believe that ML models are better trained from pairwise comparison between clusters because it better mimics human determination of semantic similarity and difference. Furthermore, ML models are able to associate combinations of similarity thresholds in particular clusters with the duplicate label as opposed to the naive approach pair-wise comparison of all words.

7.1.4 Intersection

In our third approach, we developed a unique approach to determining unique semantic content from questions. Although discussed briefly above, we believe this technique is a useful utility in feature extraction for duplication detection. To recap, rather than using TF-IDF to determine keywords, we compared only the unique words within each sentence. Initially, each question-set contains all of the words in the question. After calculating the intersection set of words between both questions, we subtracted the intersection set from both of the original question sets. The resulting question-sets are treated as keywords.

We believe that isolating the unique words in each question and comparing them effectively minimizes "false positive" results in textual analysis. If two questions are duplicates, the words retained after removing the intersection ought to be similar between both question sets. On the other hand, if they are not duplicates, the words retained ought to to indicate semantic difference. Although our current results did not demonstrate the robustness of this tool in determining "true" duplicates, they correlate to an accurate identification of "true" non-duplicates. Comparing the unique words in each question seems to have more power in determining semantic difference as opposed to semantic similarity. In our current implementation, when considering a pair of duplicate questions, we compare two sets of words with similar semantic content. However, I believe our aggregation of vectors as opposed to doing intelligent pairwise comparisons reduces the associations the ML model can make from the feature set.

We believe that integrating TF-IDF and clustering could be a significant boost to the improvement of this method. Currently, our implementation aggregates all of the vectors in each resulting question-set and computes a similarity comparison between them. However, this approach may have relied too heavily on the accuracy of the vector space generated by word2vec. We did not consider that there might be a large amount of noise generated from syntactically distinct questions. Generating a single similarity metric between the average vectors for each question set before and after subtracting the intersection extract enough data for the ML models to effectively create associations between feature distributions and classifications. Extracting all of the pair-wise comparisons between retained terms in the same clusters which have a sufficiently "unique" IDF score into a feature set may prove to be a useful future step.

As discussed above, our third approach had significantly lower recall rates than the first two approaches: the percentage of entries the model labeled as duplicates, which were "true positives" out of all of the entries with a ground truth of 1. This means that it had a greater number of "false negatives" than the other two approaches and is more likely to identify a duplicate as a non-duplicate. This may have resulted from the retainment of words with low TF-IDF values within the question sets, even after the intersection has been subtracted. These commonplace words tend not to add significant semantic content to sentences. As a result, these words add to the noise while averaging the remaining vectors before computing the similarity difference. Furthermore, it adds noise to the feature-set and prevents

ML models from as effectively creating associations between distributions of features.

However, our third approach has precision levels which are similar to the first two approaches. Our methodology is able to identify the same percentage of "true positive" duplicates out of all the model's duplicate labels. In other words, it minimizes "false positives" as much as the other two approaches. This seems to indicate that removing the intersection of shared words from both question set effectively secludes the most relevant terms for determining semantic value. Considering question sets without their intersection is a useful tool in increasing recall rates for textual analysis because it accounts for the terms in each question which uniquely determine its semantic content. This methodology minimizes the number of non-duplicate questions which are labeled as duplicate. Minimizing precision is important for Quora because classifying two non-duplicate questions as duplicates will create error in the database. Users will find irrelevant information for their questions, which is as bad, if not worse, than having to browse multiple pages to find their desired knowledge.

Extracting features from this methodology is extremely scalable in comparison to the other two approaches. The first two approaches require iterating through every word in a question to determine if it qualifies as a keyword. As our results demonstrate, subtracting the intersection set from the original question sets allows ML models effectively determines "true" non duplicate pairs. Doing a pairwise comparison between remaining words in the question sets effectively isolates all the potential words which can uniquely influence the semantic value of the sentence to be different from the other. This significantly improves run-time by reducing the number of pairwise comparisons and saving computational complexity by detecting "true negatives" early.

In our implementation, we concatenated feature extraction before and after removing the intersection set from the question sets into a unified feature set to train ML models effectively. We believe this approach allows models to create associations between distributions of the features extracted from the original question sets, distributions the features extracted from comparing unique words, and the classification label. We believe that a synthesis of the previous three tools can create a robust, efficient framework for duplicate detection within the Quora dataset.

7.1.5 Doc2Vec

In our final approach, we developed a naive approach which seemingly contradicts our previous analysis about the importance of deep-learning and the extraction of a multitude of features for ML models to train from. We fed in questions one by one to a document vectorizer and computed a similarity between vector-pairs. Our approach simply labeled all pairs with a similarity score over a specified threshold as duplicates.

Our fourth approach facilitates scalability, because it only requires converting all of the individual questions in the document to vectors and choosing choosing a number of clusters so that the documents in two different clusters will have similarities less than the provided threshold. Although this significantly reduces the number of pairwise comparisons required in comparison to the other three approaches, document vectorizers are also computationally expensive and require converting sentences into multi-dimensional vetcors which can be detected early through the third approach. Furthermore, there is a trade-off between this approach and the effectiveness of running ML models because there are not enough features extracted. The model can only draw linear associations.

Although this approach provided the best results, without extracting a greater number of features from the question-pair and including the label, machine learning models will not be able to improve upon the results from this approach. Comparing the vector similarity between the two questions has a determined cap in identifying duplicate questions; it should be concatenated into a feature set with other features which provide insight on the semantic similarity between the questions.

7.1.6 Feature Extraction

Although we experimented with feature aggregation, we did not observe a significant difference in the performance of our models. As a result, we displayed the full set of features extracted from each of our approaches.

Utilities

However, our implementation of this project contains an extensible module for developers to extract features from the Quora dataset into a custom CSV. We hope that this tool can prove facilitate deeper feature extraction in the future by promoting developer convenience. This module allows developers to efficiently customize feature extraction through stand-alone parsing, formatting, and similarity methods and utilities. For example, we were able to efficiently generate feature sets over all of the listed similarity metrics in each of the formats listed without forcing developers to write redundant code (20 feature-sets). We utilized static functions in order to ensure that the modules are stand-alone and do not have unnecessary dependencies. We believe that this module is extensible enough for developers to adapt for textual analysis outside the Quora dataset.

7.1.7 Similarity Measures

Although we experimented with similarity metric in calculating similarity scores, the highest levels of performance were observed from pairwise computations resulting from cosine similarity (that is why all of our results only consider that variation). This makes sense because the cosine similarity metric was designed for multi-dimensional vector analysis.

Utilities

However, our implementation of this project contains an extensible utility for developers to efficiently call as a static object when calculating similarity comparisons between multi-dimensional vector. This utility object contains static methods which developers can call to calculate any of the similarity metrics discussed above. We hope this utility will benefit researchers in the future by providing a modular and extensible class to quantify similarity between multi-dimensional vectors.

7.1.8 Machine Learning Algorithms

The Random Forest algorithm returns the highest precision, recall, and accuracy values on average, in comparison to any other Machine Learning model over all four approaches (excluding the naive threshold based approach). However, in our fourth approach (Doc2Vec), the Neural Network algorithm has a higher recall rate than Random Forest: .79 versus .36. In our third approach, the Neural Networks algorithm had a better performance than the K-Nearest-Neighbors algorithm.

7.2 Team-Member Contributions

Amir: Vectorization of words and sentences (word2vec and doc2vec)

Andrew: Training Machine Learning Models

Srikar: Feature Extraction

Usama: Clustering

References

[1] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[2] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." In Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp. 1188-1196. 2014.

[3] https://deeplearning4j.org/

[4] Iyer, Shanker, Nikhil Dandekar, and Kornel Csernai. "First Quora Dataset Release: Question Pairs." Data@Quora. Quora, 24 Jan. 2017. Web. 5 May 2017. <https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>.

[5] Dandekar, Nikhil. "Semantic Question Matching with Deep Learning." Data@Quora. Quora, 13 Feb. 2017. Web. 5 May 2017. <https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning>.

[6] D'Angelo, Adam. "Quality and Duplicate Questions." Data@Quora. Quora, 9 Dec. 2014. Web. 5 May 2017. <https://blog.quora.com/Quality-and-Duplicate-Questions>.

[7] Quora. "Why Quora Exists." Quora. Quora, n.d. Web. 5 May 2017. <https://www.quora.com/about>.