Duplicate Question Detection

Darpan Dodiya dpdodiya@ncsu.edu
Mohit Vinchoo mvincho@ncsu.edu
Shantanu Sharma ssharm34@ncsu.edu
Shrijeet Joshi sjoshi22@ncsu.edu

Abstract

Every Q&A website faces the problem of duplicate questions. This is especially true when the number of questions increase. This project concentrates on identifying a pair of questions as duplicate by applying various NLP classification techniques. Multiple approaches have been explored, starting with Cosine Similarity between two questions as a baseline measure. The baseline was improved by Tokenization, Stemming and Lemmatization. In addition, Semantic Similarity via WordNet, SVM (Support Vector Machine), Decision Tree, Siamese Manhattan LSTM Recurrent Neural Network approaches were explored too. Classifiers will be tuned via Grid Search and cross validation will be used to find best of them. Accuracy, precision, F1 measure and recall are the evaluation metrics. The best classifier will be reported after tuning and evaluation of all the above approaches.

1 Introduction

On every Q&A platform, multiple questions with the same intent can cause seekers to spend extra time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question.

This project to detect duplicate questions can broadly be classified into 4 main components

1. Data pre-processing
2. Feature engineering
3. Building and training models
4. Performance evaluation

1.1 Problem Statement

Given a pair of questions,identify whether or not they have the same intent. Manual identification of duplicate question pairs may work for a small number of questions, however as the complexity and scale increases, this becomes practically impossible. Identifying duplicate questions will make it easier to find high quality answers to questions resulting in an improved user experience. This will enable higher user satisfaction and an improved engagement. Another motivation is that this will reduce the role of admins and moderators on the Q/A platform.
This project will use a data-set provided by online Q&A platform Quora to build a classification model to identify duplicate questions. We evaluate different techniques to identify duplication in questions at scale.

1.2 Related Work

Since the focus of the project was to target online forums, detecting Semantically Equivalent Questions in Online User Forums (2) gave a good head start. Post pre-processing, for representation of the questions GloVe: Global Vectors for Word Representation (3) served as a good reference point for representation of words in a sentences as a vector. The ideas proposed in From Word Embeddings To

Document Distances (4) helped on leveraging distance measures to generate a baseline classification model. (5) Gave a Siamese adaptation of Long Short-Term Memory(LSTM) to assess the semantic similarity between sentences.

2 Method

The data-set of question-pairs is given as tuples of questions and labels. The value of label is 1 for similar questions and 0 otherwise. In order to feed data to the classifier we needed to represent the data into format that the classifier can understand. We have tried an iterative approach to improve accuracy while adding features and semantic understanding. Details of each method are given below starting from baseline cosine similarity

2.1 Baseline : Cosine Similarity

The first, naive approach towards identifying question pairs is to split each sentence into words and then find cosine distance between the pair. This was done to explore if cosine similarity is a good measure of duplicity.The cosine similarity can be calculated by:

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\parallel \vec{x} \parallel \parallel \vec{y} \parallel} \tag{1}$$

where x and y are vectors of words of the question pair. This is a naive approach without any pre-processing or cleaning and is used as a baseline.

2.2 KNN

As an extension of the baseline, we compute cosine distance between the two questions after data has undergone NLP cleaning tasks and run KNN. In this experiment we will be using count vectorizer and ngrams of the questions. Ngrams are useful as they help us understand positioning of words for example two words "not" and "good" when considered together give an entire new meaning. An important thing to note is that the classifier takes a tuple of (cosine similarity, label) to train. i.e. we converted our questions data into a single feature. Experiments with values of k showed that we get best results with the hyper-parameter k=2.

2.3 Semantic Similarity WordNet

WordNet is a huge library of synsets for almost all words in the English dictionary. The synsets for each word describe its meaning, part of speeches, and synonyms/antonyms. The synonyms help in identifying the semantic meaning of the sentence, when all words are taken together.

In our earlier approach of finding cosine distance between questions, there was lack of semantic understanding – There might be two questions with a high percentage of common words, but different meanings. Approach described in paper (6) eliminates this by considering semantic meaning of words. This approach operates by representing each sentence as a binary vector (with elements equal to 1 if a word is present and 0 otherwise), a and b. The similarity between these sentences can be computed using the following formula:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a}\mathbf{W}\vec{b^T}}{|\vec{a}||\vec{b}|} \tag{2}$$

where W is a semantic similarity matrix containing information about the similarity of word pairs. This approach resulted in an improvement over the naive cosine similarity and 2-NN.

2.4 Siamese Manhattan LSTM

Siamese networks are a type of network that have two or more identical sub-networks in them. They have good performance when we need to check the similarity of 2 input items. This model converts the given input sentences to its vector representation. This is used in the final hidden layer. We use the

similarity between these representations as a predictor to determine if the input sentences are similar [5]. Here we have used Manhattan distance to check the similarity of the sentences (MaLSTM). We have used Google's word2vec Embedding to get the words for the deep learning network. In the embedding layer, the words from the above vectors are selected and looked up into the corresponding embedding. This is the main layout of this model.

3 Experiment

3.1 Hypothesis

Hypothesis 1: Duplicate questions have more cosine similarity than non-duplicate questions: The median of cosine similarity scores of non-duplicate questions is .48 where as the median duplicate questions is .69. This shows that higher the cosine similarity is present in questions that are similar.

Hypothesis 2: Removal of stop words with negative connotation: Removal of all stop words also removed words such has "not" "no" etc. these words make a lot of difference in our problem since "this is spring" "this is not spring" will be same vectors if we blindly remove all stop words including "not" and hence labeled incorrectly.

Hypothesis 3: Usage of N-grams leads to better results: Simple vectorization does not take into account the order of words but ordering of word plays an important role in understanding the question, hence using Ngrams should improve the accuracy our predictions.If we use observations of hypothesis 2 and add 2Grams while vectorizing the questions we found that test accuracy increases from 61% to 64%

Hypothesis 4: Semantic meaning gives a better prediction than only syntactic comparison: Cosine similarity only considers the word vectors between two questions and computes a simple cosine distance between the 2 vectors it does not do any semantic analysis. If the algorithm tries to understand the underlying semantic structure of the question we should be able to get better results. We achieved this by using WordNet and later by using google word embedding while using LSTM. The results were better, we were able to increase test accuracy to 73% with WordNet and 82% with MaLSTM network which uses Google word embedding and the Adadelta gradient descent optimizer.

3.2 Python environment and packages

We have used Python and other supporting packages for our analysis. Notable components of the environment have been described below:

Table 1: Environment details

Package/Software	**Version**
Python	v3.7.2
Jupyter Notebook	v5.7.6
NLTK	v3.4
Numpy	v1.15.1
Pandas	v0.23.4
Scikit-learn	v0.19.2
matplotlib	v2.2.3
TensorFlow	v1.5

3.3 Dataset

The data of duplicate questions pairs has been provided by Quora on Kaggle platform. (1) The ground truth of the dataset has been supplied by human experts on Quora. Since Human labeling is not accurate, the labels are not believed to be 100% accurate.

Table 2: Data fields

Attribute	Description	Example
id	the id of a training set question pair	1
qid1, qid2	unique ids of each question	1, 2
question1, question2	the full text of each question	"How are you?", "Are you good?"
is_duplicate	the target variable, 1 if duplicate, 0 otherwise	0

Table 3: Data summary

Type	Record Count
Total question pairs	404290
Duplicate question pairs	149263
Non duplicate question pairs	255027
Missing values	3

3.4 Experiment design

We have implemented a pipeline of operations in order to perform experiments for a classifier. The pipeline was more or less same but few subtle changes were done in order to improve the accuracy in each experiment. The general flow and steps are outlined in the Figure 1.

Figure 1: Flow of execution

3.4.1 Pre-processing

Every experiment that we have performed involved pre-processing of data. This step is applicable across all the experiments. The data pre-processing step started with eliminating records with duplicates and missing value. Next we eliminated non-alphabetical characters and then tokenized the questions. Our analysis showed that stop words in the questions did not add value therefore we also eliminated stop words.
After this we performed the NLP tasks of stemming and lemmetization. Once we were done with this we converted the questions to vectors using the Count Vectorizer. While performing KNN we used n-grams to perform count vectorization.

3.4.2 Baseline

As described in *Hypothesis 1*, we believed that a measure as simple as cosine similarity between two questions would serve as good indicator of baseline performance of the models. Using CountVectorizer of scikit-learn, we vectorized the sentences and calculated similarity.

3.4.3 Improving baseline

As we proved that a simple measure as cosine similarity could work well, we tried to improve baseline. The experiments included removing stop words, removing non-alpha numeric characters, removal of stop words with negative connotation. (*Hypothesis 2*). We also tokenized questions. NLTK package was extensively used to perform these experiments. This experiment improved baseline by 4-5%.

3.4.4 Semantic meanings, N-grams, KNN

Once we have baseline ready and also pre-processed data, we experimented with different approaches such as KNN, WordNet (*Hypothesis 4*), N-grams (*Hypothesis 3*) to see which could give best results.

3.4.5 Long Short-Term Memory Neural Network

In addition, artificial recurrent neural networks were explored in further experiments, such as Siamese Manhattan LSTM Recurrent Neural Network. A preliminary experimentation based has been done, more rigorous experimentation is underway.

4 Results

Below are partial results from our experiments so far. As of now, we have only taken accuracy in consideration for the model evaluation. Later on, the models will be evaluated on other parameters such as F score, recall, precision. This section is under construction and the results may change as we revise the experiments. For LSTM we will be cross validating on several epochs. In our rough

Table 4: Experiment results

Experiment	**Accuracy**
Cosine Similarity	63%
WordNet Semantic Similarity	72%
2-NN Classifier	63%
LSTM Neural Network	82%

implementation every epoch is taking around 10-12 minutes. Best results as reported above have been seen on number of epochs in the range [20 - 25].

5 Conclusion

We have explored multiple methods to tackle the problem of detecting Duplicate Questions as shown above with varying performances. Further, we will work on optimizing these methods as shown below.

6 Future plan

Below is a list of experiments that we plan to do in the coming days.

1. Feature Engineering : We have till now used the question pairs to generate a new feature cosine similarity, we are looking to explore if we could use more features such as similar nouns and verbs. This will require in-depth data exploration.
   Team Member - Darpan Dodiya will work on this.
2. Classify with SVM : We want to use SVM and test the accuracy.
   Team Member - Shantanu Sharma will work on this.
3. Using Ensemble Methods XGBoost : Does usage of ensemble method give us a better result than what we have achieved.
   Team Member - Shantanu Sharma will work on this.
4. Exploring evaluation criteria : As of now we are using accuracy as evaluation criteria, we wish to explore if F1 score could be a better indicator of our results. Also we will use k-fold cross validation method to rank classifiers.
   Team Member - Mohit Vinchoo will work on this.
5. For LSTM, we will complete the implementation within first week of April. We shall explore other Gradient descent optimizers, work with more data and research methods to optimize our model further.
   Team Member - Shrijeet Joshi will work on this.

References

[1] https://www.kaggle.com/c/quora-question-pairs/data

[2] Bogdanova, Dasha et al. "Detecting Semantically Equivalent Questions in Online User Forums." CoNLL (2015).
https://aclweb.org/anthology/K15-1013

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
https://nlp.stanford.edu/pubs/glove.pdf

[4] Matt J. Kusner , Yu Sun , Nicholas I. Kolkin , Kilian Q. Weinberger, From word embeddings to document distances, Proceedings of the 32nd International Conference on International Conference on Machine Learning, July 06-11, 2015, Lille, France
http://proceedings.mlr.press/v37/kusnerb15.pdf

[5] Jonas Mueller. Aditya Thyagarajan, From Siamese Recurrent Architectures for Learning Sentence Similarity, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).
http://www.mit.edu/~jonasm/info/MuellerThyagarajan_AAAI16.pdf

[6] Fernando, Samuel Stevenson, Mark. (2009). A Semantic Similarity Approach to Paraphrase Detection. Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics.