proposal

November 27, 2017

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Djona Fegnem

1.2 Proposal: Identify question pairs with the same intent.

1.2.1 Domain Background

Websites such as StackOverflow, Reddit, Quora provide a platform for users to ask and answer questions. The democratization of the Internet has enabled an increase engagement on such sites. StackOverflow since inception has roughly 13 million questions, 21 million answers, 56 million comments, and 7 million users as of April 1st, 2017. With this growing number of users and questions, some (websites) have started looking for ways to remove redundancies among questions.

One way to achieve such task is to limit the number of repeated questions. StackOverflow for instance, has tried to solve this problem by relying on its users. An attempt to post a question on StackOverflow provides the user with a search tool which allows him/her to find semantically related questions prior posting one with a set of guidelines on how to do so effectively. This first approach has some shortcomings: Firstly, the search functionality provided usually performs query search based on questions' word syntax - they tend to match questions with the same word form. Secondly, people use different vocabulary - users in different contexts, or with different needs, knowledge or linguistics habits will describe the same information using different terms. Finally, some users usually skip those guidelines and don't use the search tools. In addition to the search tool, StackOverflow has a duplicate questions policy, which allows its users to flag or vote to close duplicates.

Given these limitations in the existing tools and mechanisms, can it be a better alternative?

With the growing number of data available on those sites, it becomes apparent that machine learning can be a solution. Machine learning enables computers to learn patterns from data thus, can learn patterns from data generated from users' interactions and provide a model that can automatically detect repeated questions. This model can then be integrated into a search tool such that when a user attempts to ask a question, the search tool automatically searches for duplicates and provides an answer if it exists. This will improve the user experience and will allow for an efficient use of computer resources.

In a recent Kaggle challenge, Quora has challenged the data science community to solve this task using their data. In [paper 2], the authors have used an artificial neural network technique called convolution neural network (CNN) to solve similar problem. In this capstone project, we

will apply the same technique to solve our problem. Our neural network will be trained using the Quora published question pairs dataset.

1.2.2 Problem Statement

Given a pair of sentences, can we build a model able to identify whether or not sentences within the pair are semantically equivalent? By semantically equivalent we mean sentences or questions where the answers are the same.

1.2.3 Datasets and Inputs

The dataset used for this project is the Quora published question pairs dataset on Kaggle. Our dataset has 404,290 data points with the following features:

- Id: a unique identifier for each entry.
- qid1: an identifier for the first question in a pair.
- qid2: an identifier for the second question in a pair.
- question1: the first question in a pair.
- question2: the second question in a pair.
- is_duplicate: a value 1 or 0 that indicates whether the questions in a pair are duplicate or not.

1.2.4 Solution Statement

We have a supervised learning task more specifically, a classification problem. Our goal is to build a classifier able to evaluate the similarity between two sentences. Since we have texts, we will need to perform feature engineering prior training. During feature engineering, we will capture the meaning of sentences via low dimensional vectors also called word embedding. We will explore two word embedding techniques. One technique based on term co-occurence matrix (Latent Semantic Analysis - LSA)[paper 4], another technique based on neural network (word2vec)[paper 3]. This first step will give us feature vectors that will be used to train our model. During training we will evaluate two approaches: In the first approach (baseline), we will use the embeddings from LSA to compute the cosine similarity score between sentences and use it to train our model. Our algorithm will be an ensemble learning algorithm called eXtreme Gradient Boosting (XGBoost). In the second approach we will train a deep neural network using pre-trained word2vec embeddings, the output will be compared against a threshold that we will choose during training. Our evaluation metric will be the f1 score.

1.2.5 Scoring method: Cosine similarity

For scoring method we will be using the cosine similarity. The cosine similarity measures the proximity of two vectors in the vector space.

$$sim_{q,d} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sum_{i=1}^{|V|} q_i^2 * \sum_{i=1}^{|V|} d_i^2}$$

where:
$sim_{q,d}$ = the cosine similarity of the document q and d.
$|V|$ = the size of the vocabulary.

q_i = the tf-idf weight of term i in the document q.
d_i = the tf-idf weight of term i in the document d.

1.2.6 Benchmark Model

Our baseline model will consist of the following steps: we will perform feature engineering on questions within a pair, and then we will apply XGBoost on the new dataset. Our feature engineering task will involve using an LSA model to extract questions' vector representation and computing the similarity score between questions of a pair. Weights in the term-document matrix used to build the LSA model will be the Term Frequency-Inverse Document Frequency (TF-IDF)value between the document and the term. The similarity score, together with its corresponding class(duplicate 1, or no duplicate 0) value will form our new dataset.

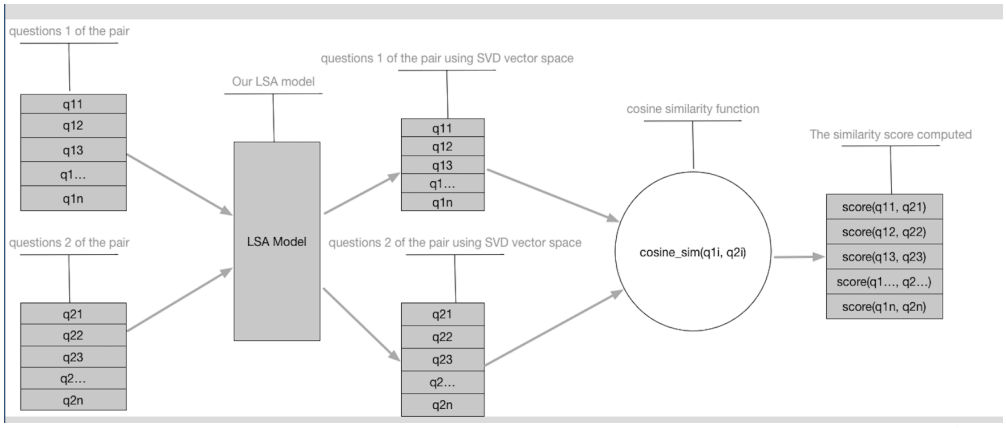
Term-document matrix with TF-IDF A term-document matrix represents words or terms based on documents they contain. Values in the matrix can be either the word count, or the term frequency inverse document frequency(TF-IDF). The TF-IDF captures both the frequency of terms within the document and its rarity in the corpus.

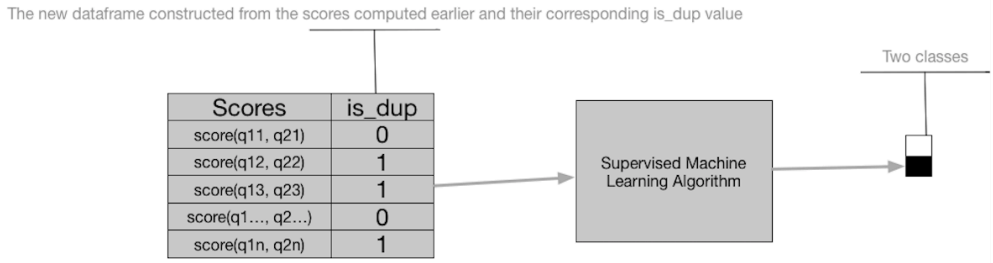
$$w_{t,d} = (1 + \log_{10} tf_{t,d}) * \log_{10}(\frac{N}{df_t})$$

where:
$w_{t,d}$ = the tf-idf weigth of the term t in the document d.
tf_t, d = the term frequency of t in in the document d.
df_t = the document frequency of the term t.
N = the total number of document in the corpus.

We will use term-document matrix with TF-IDF weights during the modelling of our benchmark. Each document or question will consist of a TF-IDF vector in R^|V|, with |V| being our corpus' vocabulary size. TF stands for term frequency, it is the number of time a particular vocabulary appears in a document. IDF stands for inverse document frequency, it is the inverse of the number of documents in the corpus that contains a particular term.

Baseline Feature engineering Our baseline feature engineering will consist of the following: we will build our LSA model using questions in columns "questions1" and "questions2" as our set of documents. The weights in our term-document matrix will consist of the TF-IDF value between the term and the document. Next, we will extract the vector representation of each question. Finally, We will compute the similarity of each pair of questions using cosine similarity.





1.2.7 Evaluation Metric

For this project, we will use f1 score as evaluation metric.

$$f1_{score} = 2 * \frac{precision * recall}{precision + recall}$$

$$precision = \frac{true_{positive}}{true_{positive} + false_{positive}}$$

$$recall = \frac{true_{positive}}{true_{positive} + false_{negative}}$$

where:

$true_{positive}$ = the number of similar pairs of sentences our algorithm has successfully identified.

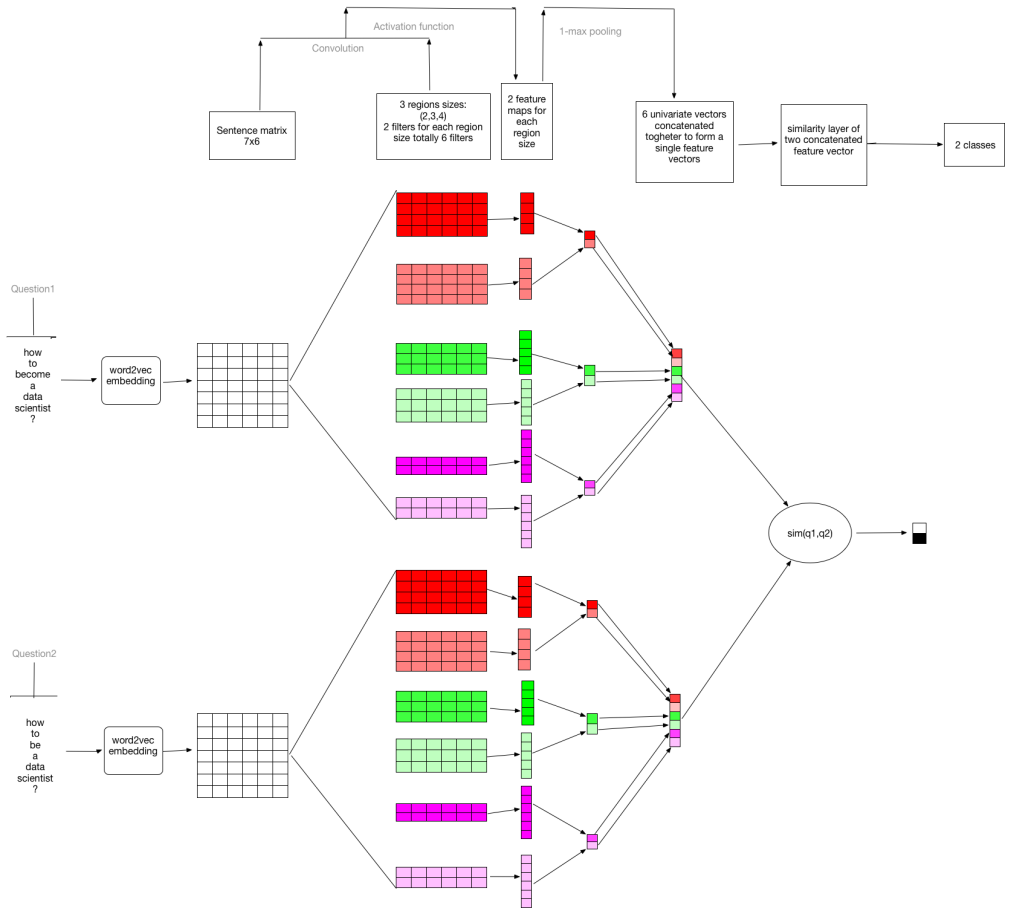
$false_{positive}$ = the number of similar pairs we have identified as non similar.

$false_{negative}$ = the number of non similar pairs we have identified as similar.

The precision measures the proportion of positive instances correctly classified out of all the one returned by our model during evaluation. The recall measures the proportion of positive instances correctly classified out of all the positive instances returned by our model during our evaluation. f1 score is a single metric that captures both precision and recall.

1.2.8 Project Design

We will first perform an analysis of our data. It will involve removing any anomalies such as missing values, and non-alphanumeric characters. Then we will build and evaluate our benchmark model. After that, we will build and evaluate the model architecture described in the diagram below. It is a convolutional neural network (CNN) inspired from both [paper 5] and [paper 2]. The input is the tokenized text of questions within a pair, the output is the similarity score. The CNN first transforms words into real-valued feature vectors or word embeddings. Next, the convolutional layer is used to construct two distributed vector representations, one for each input question. Finally, the CNN computes the similarity score between them. Pairs of questions with a similarity above the threshold (hyperparameter that we will define during training) are considered duplicates.



1.2.9 Software requirements

1. python 3.5.3.
2. gensim to build our word embedding.
3. tensorflow 1.1 : to build our neural network.

1.2.10 References

1. Quora dataset.

2. [paper 1] - Convolutional Neural Networks for Sentence Classification, Yoon Kim, New York University.
3. [paper 2] - Detecting Semantically Equivalent Questions in Online User Forums, Dasha Bogdanova, C cero dos Santos, Luciano Barbosa and Bianca Zadrozny ADAPT centre, School of Computing, Dublin City University, Dublin, Ireland
4. [paper 3] - Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Google Inc., Mountain View, CA
5. [paper 4] - Indexing by Latent Semantic Analysis, Scott Deerwester, sept 1990.
6. [paper 5] - A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification, Ye Zhang, Byron C. Wallace.