Question Similarity in Community Question-Answering

SHERLY sherly

December 2018

Abstract

The objective of this project is to study the impact of syntactic, semantic and thread-based features on community question answering similar questions detection. The data used in this study is from SemEval-2017 Task 3: Community Question Answering Subtask C - Question-Question similarity.[1]. The task is approached as a binary classification task as well as a classification task with three classes, *PerfectMatch*, *Relevant* and *Irrelevant* as provided in the original dataset. The word embedding for the task is generated with several methods including using pre-trained vectors. The sentence embedding are generated with weights of its subject and body as well as evaluating it on important sentences.

Keywords: *NLP, similarity, SemEval*

1 Introduction

Today, text similarity detection is one of the most common task in Natural Language Processing (NLP). There has been different results for different text data generated in each domain and in this project, we will be studying in more detail in the domain of Community Question Answering.

The report is structured in 6 main sections. Section 2 describes the task that is being proposed in SemEval Task 3. Section 3 describes the baseline approaches to the text similarity task. Section 4 describes the each of the different proposals to approaching the tasks in more details. Section 5 describes the experimental setup and section 6 studies the experimental results.

2 Task Description

A question to question similarity problem consists of four parameters, which can be defined as a tuple of elements (S, B, Q, y). $S = [s_1, s_2, .., s_m]$ denotes the subject of a related question, where each s_i is of the dimension equals the size

[1] `http://alt.qcri.org/semeval2017/task3/`

of the embedding that represents it. Similarly, $B = [b_1, b_2, .., b_m]$ denotes the body of a related question. The set of related questions are retrieved by a search engine and m denotes the size of set of the related questions. $Q = [q_1, ..., q_n]$ is the original questions that for which related questions are being looked for and n denotes the number of such questions. y Y is the label representing the degree to which the proposed question is similar to that question. Y = PerfectMatch, Relevant, Irrelevant where PerfectMatch indicates that the question is a good match to the new question, Relevant indicates the answer are potentially similar to that question and Irrelevant is not a match or similar to the new question. Given S, B, Q, the task in this project is to rank the labels assigned to each question based on the conditional probability P(y | S, B, Q).

3 Baseline Approaches

In this section, description of the implemented baselines will be discussed. Two baseline approaches have been implemented with different word embedding namely (i) TF-IDF (ii) Word2Vec with sentence embeddings.

The sentence embedding approach consists of representing each question (original or related) by an embedding vector. The embedding vector is the average of the vector embedding of each word of the question. To determine the similarity of the pairs of questions, the cosine similarity is computed. The related questions are ranked according to their scores regarding the original questions. Stop-words are removed from each sentenced and only words of frequency more than 1 is being kept.

4 Proposal

The exploration to generate the best representations to rank the question-question similarity are performed in 4 dimensions (i) word embeddings (ii) document embeddings (iii) features (iv) ranking methodologies.

In the first dimension relating to word embeddings, explorations of different methods to generate the word representations will be performed in order to study the effect of these different methods on the text data in the context of community question answering. Five different ways of generating the words embeddings are being explored (i) TF-IDF (ii) Word2Vec (iii) Latent Semantic Indexing (LSI) (iv) Skip-Gram and (v) Pre-trained model from Google News.

In the second dimension of document embeddings, three different variations of sentence embeddings will be explored (i) equally weighted subject and body (ii) weighted subject and body and (iii) weighted embeddings based on sentence selection.

In the third dimension, feature engineering is being performed in order to generate more useful features that will enhance the ranking predictions.

Lastly, the ranking problem will be approached in two methods (i) binary classification and (ii) multiple class classification.

4.1 Word Embedding

Evaluation of the similarity scores based on the different types of word embedding for each classes. Generating embedding for each question based on text selection using graph-based method. Generating embedding by weighting subject and body embedding.

4.1.1 TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) is used to produce a weight for each word in each document i.e. each question. To compute the weight of each term:

$$W_{ij} = TF_{ij} + IDF_j$$

where

W_{ij} = weight of term in T_j in document D_i,

Tf_{ij} = frequency of term T_j in document D_i, i.e. (number of times term T_j appears in a document D_i) / (total number of terms in the document D_i),

$IDF_j = log$(Total number of documents / Number of documents with term T_j in it).

TF-IDF assigns to term T_j a weight in document D_i such that it is,

- high when a term T_j occurs many times within a small number of documents
- lower when a term T_j occurs fewer times in a document or many times in many documents
- lowest when the term occurs in almost all the documents.

As a baseline approach, we compute the TF-IDF of each term and use it as the word representation vector.

4.1.2 Word2Vec

Word2Vec model proposed by Mikolov et al, 2013[4] is being used to generate the word embeddings with two different architectures, skip-gram and continuous bag of words (CBOW).

Skip-Gram which is slower performs better for infrequent words and CBOW is fast. There are two training algorithms, hierarchical softmax which performs better for infrequent words and negative sampling which performs better for frequent words. Subsampling of frequent words are applied in order to improve both accuracy and speed in the event of large dataset. The dimension of the words chosen is 300 in order to standardize the dimensionality of vectors generated in the different approaches. Another hyperparameter is the window size for word context.

Two models are trained with the following hyper-parameters:

Model	Architecture	Training Algo	Sub-Sampling	Dim	Window Size
1	Skip-Gram	Hierarchical Softmax	1e-5	300	10
2	CBOW	Negative Sampling	1e-5	300	5

4.1.3 Latent Semantic Indexing

Latent semantic analysis analyses the relationship between documents and the terms in it with the concept that words that have similar meanings will appear in similar documents. A matrix containing word counts per document in the corpus is is generated and a dimensionality reduction method called Singular Value Decomposition is applied on the matrix while preserving the similarity structure. To compute the similarity between words, we take the cosine of the angle between the two vectors. A value close to 1 denotes similar. Implements fast truncated SVD (Singular Value Decomposition). The library gensim implements LsiModel with fast truncated SVD.

4.1.4 Skip-Gram

Mikolov et al, 2017[1] proposed a new approach which represents words as a bag of character n-grams and trained with the skip-gram model. Vector representation is generated for each character n -gram and thus a word is being represented as the sum of the character representations. By using this approach, it allowed sharing of representations across words and hence are able to learn better representation for infrequent words.

The model is used to train on the training data provided for SemEval Task 3 in order to generate the context specific representation.

4.1.5 Pre-trained Google News Vectors

Google has published pre-trained vectors trained on part of Google News dataset. The model contains 300-dimensional vectors for 3 million words and phrases. The representation is trained with Word2Vec approach with continuous bag of words architecture. The hyperparameters are sub-sampling with threshold of 1e-5 and negative sampling with 3 negative examples per each positive one.

The pre-trained vectors is used a transfer learning of the google news dataset on the community question answering to explore the impact.

4.2 Document Embedding

4.2.1 Subject and Body

To generate the representation of the question, the vector representation of the subject and body will be concatenated.

4.2.2 Subject-Body

Wu et al, 2018 [6] proposed a model where question subject is the primary part of the question representation, and the question body information is aggregated based on similarity and disparity with the question subject. The model explored in this project will study the best weighting for subject body to rank the relevance of the questions. It follows:

$$score = alphaxsimilarityofsubject + (1 - alpha)xsimilarityofbody.$$

The subject of the question is the summary of the question body which highlights the important keywords to the question and hence it will provide more information to the similarity of the question.

4.2.3 Sentence Selection

Barrón-Cedeño et al., 2017 [5] proposed text selection method for community question answering which seems to perform better for Subtask A and C for Community Question Answering task but not the Subtask B studied in this project. An alternative method for text selection is utilised in this project in order to further study the impact of text selection using the graph-based method, LexRank to retrieve most importance sentences in a text and utilise it to generate the document representation for a question. LexRank computes sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. A connectivity matrix based on intra-sentence cosine similarity is used as the adjacency matrix of the graph representation of sentences. A cluster of documents can be viewed as a network of sentences that are related to each other. Some sentences are more similar to each other while some others may share only a little information with the rest of the sentences.Sentences that are similar to many of the other sentences in a cluster are more central (or salient) to the topic.

4.3 Features

In the feature engineering step of this project to incorporate more measures of similarity into the ranking tasks, features proposed by Filice et al, 2017 [2] are generated in addition to several other measures.

- Lexical Cosine similarity, Jaccard coefficient (Jaccard, 1901) and containment measure (Broder, 1997) of n-grams of word lemmas (n = 1; 2; 3; 4 was used in all experiments); Longest common substring measure (Gusfield, 1997), Longest common subsequence measure (Allison and Dix, 1986). In addition, Word Mover's distance, Canberra distance, Cityblock distance, Euclidean distance, Minkowski distance and Braycurtis distance are also computed as features. Token ratios are also computed as part of features.
- Semantic Similarities: Cosine similarity between additive representations of word embeddings generated by applying word2vec (Mikolov et al., 2013)

to the entire Qatar Living corpus from SemEval 20153. Five features are derived considering (i) only nouns, (ii) only adjectives, (iii) only verbs, (iv) only adverbs and (v) all the above words.

4.4 Task Approaches

Two methodologies are used to rank the questions (i) binary classification with Random Forest Classifier (ii) binary classification with XGBoost and (iii) multiclass classification with fastText.

4.4.1 Classification with Random Forest Classifier

A baseline approach to classification is performed with Random Forest Classifier in order to compare the performance of the other approaches.

4.4.2 Classification with XGBoost

XGBoost, eXtreme Gradient Boosting model is based on gradient boosting framework which is a technique used to build predictive tree-based models. Boosting is an ensemble method where new models are added to correct the errors made by trained models until no further improvements. New models created predicts the errors of prior models and then added together to make the final prediction. The objective of the XGBoost model is given as:

$$Obj = L + \omega$$

Where, L is the loss function to be optimized, and ω is regularization to prevent overfitting

4.4.3 Classification with FastText

Mikolov et al [3] proposed a linear model with rank constraint to perform classification

A bag of n-grams is generated for each word and it is denoted by x1, x2, .., xN. The word representations are then averaged into a text representation, which is in turn fed to a linear classifier. A softmax function f is used to compute the probability distribution over the predefined classes. The model is optimized by minimizing the negative log-likelihood over the classes.

5 Experimental Setup

5.1 Data Preprocessing

It is important to note that the text data are being pre-processed before the apply the techniques and methodologies to generate their representations. English stop-words are being removed from the text and embeddings are experimented with both applying lemmatization and without lemmatization to study the impact of lemmatization on the text data of community question answering.

5.2 Evaluation Metrics

The official evaluation metrics in SemEval 2017 is Mean Average Precision (MAP). For each of the variations of the embeddings proposed, MAP will be computed. In the classification tasks that we have proposed to classify the different questions, precision, recall and F1-score will be computed. For a perfect ranking, the model has to place PerfectMatch and Relevant questions above the Irrelevant questions and in the classification task, correct label has to be predicted.

6 Experimental Results

Non-Lemmatized Text			
Embedding	Relevant(Avg)	Irrelevant(Avg)	MAP
TF-IDF	0.117871	0.039916	15.9085
Word2Vec (Skip-Gram)	0.999636	0.999632	15.9053
Word2Vec (CBOW)	0.918288	0.907156	15.9094
LSI	0.779652	0.718705	15.7457
Skip-Gram with FastText	0.904870	0.872059	15.8759
Pre-trained Google News Vector	0.647539	0.571696	15.9034

For non-lemmatized text, TF-IDF, Word2Vec and Pre-trained Google Vectors seem to perform best. Skipgram methodologies does not seem to perform well on this task.

Lemmatized Text			
Embedding	Relevant(Avg)	Irrelevant(Avg)	MAP
TF-IDF	0.139164	0.049151	15.9152
Word2Vec (Skip-Gram)	0.999647	0.999648	15.8509
Word2Vec (CBOW)	0.940885	0.933219	15.9593
LSI	0.772477	0.713514	15.6020
Skip-Gram with FastText	0.907106	0.874882	15.7622
Pre-trained Google News Vector	0.663743	0.592109	15.9442

For lemmatized text, Word2Vec with CBOW method and Google News Pretrained vectors have the best performance.

We observe that both model are trained with CBOW and transfer learning of Google News model on Community Question Answering are performing well as compared to training on its own corpus.

Model	Embedding	Precision	Recall	F1-score
Binary Classification with RandomForest	Body only	0.6801	0.6724	0.6519
Binary Classification with XGBoost	Body only	0.7065	0.7079	0.6984
Binary Classification with RandomForest Weighted	Subject-Body	0.6899	0.6897	0.6800
Binary Classification with XGBoost Concatenated	Subject-Body	0.7337	0.7341	0.7270
Binary Classification with XGBoost Weighted	Subject-Body	0.7384	0.7391	0.7329
Binary Classification with XGBoost	Sentence Selection	0.7384	0.7391	0.7329
Multiclass Classification with XGBoost	Body only	0.6200	0.6280	0.5896
Multiclass Classification with XGBoost	Concatenated Subject-Body	0.6563	0.6554	0.6206
Multiclass Classification with XGBoost	Weighted Subject-Body	0.6593	0.6642	0.6302
Multiclass Classification with XGBoost	Sentence Selection	0.6478	0.6567	0.6227
Multiclass with fastText (text features)	Subject and Body text	0.6205	NaN	NaN
Binary Classification with fastText	Subject and Body text	0.5880	NaN	NaN

From the table above, we observe that weighting the subject and body as its representation improves the predictive power of the classification. While sentence selection may have improved the binary classification, it does not improve the multi-class classification which is also observed by Barron et al that it is yet conclusive that text selection improves the performance. Binary classification has higher performance as compared to multi-class classification which is sufficient in this case of ranking *PerfecMatch, Relevant* above *Irrelevant*. Classification with fastText with text only data does not perform as well. Adding POS-tags based features improves the classification model. Modelling the task with just its vectors similarity does not model well and hence feature engineering is an important aspect to this tasks besides improving the representation.

7 Conclusion and Future Work

There are still many study that could be done on this project to understand the implication the different components of text to this task. Some of these may include Name Entity Recognition (NER) to better identity the important

entities, chunking and even generating a better word representation that better represents the Community Question Answering domain.

References

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[2] Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 326–333. Association for Computational Linguistics, 2017.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[5] Salvatore Romeo, Giovanni Martino, Alberto Barrón-Cedeño, and Alessandro Moschitti. A multiple-instance learning approach to sentence selection for question ranking. pages 437–449, 04 2017.

[6] Wei Wu, Xu SUN, and Houfeng WANG. Question condensing networks for answer selection in community question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1746–1755. Association for Computational Linguistics, 2018.