SimQueFi: Similar Question Finder for Stack Overflow

Darshan V Ramu
Computer Science
USC Viterbi
darshanv@usc.edu

Pradeep Sridhar
Computer Science
USC Viterbi
sridharp@usc.edu

Vishal J Singh
Computer Science
USC Viterbi
vishaljs@usc.edu

1 Introduction

Every now and then, when we hit a wall with technical doubts regarding the language usage in terms of its syntax or an algorithm implementation we tend to search solutions using Stack Overflow [1]. It is one of the active and highly rated forums where we can get answers quickly for any technical question from the domain experts. One of the first things the user who is seeking answer will do is to search if the question is already asked by someone else. At this point he tries to use Stack Overflow search engine and it promptly returns a set of questions. But one huge problem with the returned result is that it contains the questions which exactly match the words in the user query. This leads to loss of information i.e., useful answers by someone else for a similar question is hidden.

We noticed that we can do better at this using the NLP techniques learnt in the course to show more relevant and already solved questions to the user based on the question's semantics rather than the exact words during search to enrich the user experience and help him find the solution quickly.

We provide our own algorithm which can be used as a plug-in to Stack Overflow. This reduces the re-post of the same questions again and again. We reduce the duplication of data as well and conserve the computational resources and storage space for Stack Overflow in this ever increasing era of data.

The work was challenging in terms of the questions written by user in new grammar and also in finding the hidden structure in the questions asked.

2 Related Work

There have been lot of work done in the field of semantic similarity and document similarity but our approach of finding similarity between questions based on synonyms for the words after considering the Parts-of-Speech (POS) tag is very unique. The problem of trying to find similar question in Stack Over-flow was not done till now and we have made a fruitful attempt to solve it. Our tool can be used as a plugin to Stack Overflow website to help users to solve their questions quickly if a similar question is previously answered. Our tool is not specific only to Stack Overflow website but can be used to solve similar similarity problems.

3 Data

The training data consisted of a set of 1800 questions from Stack Overflow website. These are actual user generated queries on Stack Overflow. We used the StackExchange API[2] for retrieving the data.

3.1 Data retrieval module

To create the data retrieval module we implemented a web page with two input options provided to the user. The user could enter the tag or some section of the question text that was desired. These parameters were then passed to the StackExchange API as a part of an AJAX request.

The API returns the result in JSON format. This JSON object was traversed on the client side to get the actual question text. The entire text was made lower case. These questions were then stored in a file in the text format.

The data retrieval module is available here [3]

3.2 Data retrieval

We collected questions across various domains using the retrieval module.
Sample tags used for retrieving data: C, Java, ADFS, JavaScript, etc.
Sample question texts used: How to get minimum, find least, restart IIS, etc.

4 Technical Approach

We wrote our algorithm to perform the following steps (3.1- 3.6) to solve the given problem of finding similar questions:

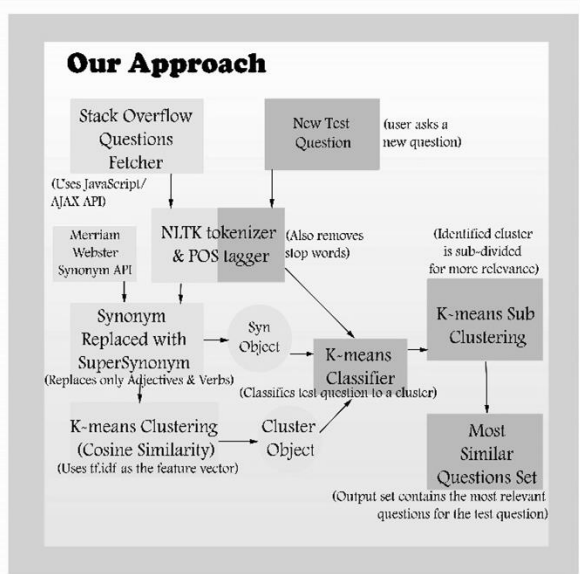

Figure 1: SimQueFi algorithm

4.1 Data Retrieval

Please refer Data Section of this report.

4.2 Data annotation and Data pre-processing:

Collected data from Stack Overflow API is tokenized using NLTK tokenizer and annotated for POS tags using NLTK [4] POS tagger. The tagged data is then processed to remove stop words using Stop Word list. Stop word list consisted of cumulative words from NTLK module's stop word list and our own word list. Our stop word list was constructed after careful analysis of Stack Overflow questions (~1800) and we found out the most frequent words used in those questions viz., what, how, to etc.

Once the question was POS tagged and the words removed according to the stop word list, they were forwarded to the stemming module which used Snowball English Stemmer [5].

4.3 Word substitution for selective words

After annotating the questions using a POS tagger, the words in the question which have been tagged as verb (VBG, VB, VBD, VBP, VBZ and VBN) [6] or adjective (JJ, JJR and JJS) [6] are considered for substitution. For these words, synonyms are obtained using Merriam Webster synonym Web API [7]. Different synonym tools like WordNet interface [8] provided in NLTK and Wordnik's [9] python interface were evaluated before Merriam Webster synonym Web API was selected. The synonyms provided by WordNet were very limited and Merriam Webster performed better when compared to Wordnik when tested for several words. The word and its synonyms obtained from Merriam Webster synonym Web API were replaced in the question with a super synonym (X-label) which was stored in a dictionary along with the words for future classification of test questions.

For example, questions like "what is the smallest element in an array" and "what is the least element in an array" will be replaced as "what is an X1 element in an array" where X1 is the super synonym (X-label) for "small" and "least". The words and their synonyms in the questions are replaced with the label to increase the likelihood of clustering questions having words with similar meanings to be clustered together in the same cluster.

4.4 Data clustering

We used K-means clustering algorithm to cluster the given data. We implemented K-means with Euclidean distance and Cosine-similarity as a measure to cluster the given input. We wrote our own cosine-similarity module to test the effectiveness of cosine-similarity measure approach before dwelling into full-fledged K-means algorithm.

We wrote a script to calculate term frequency (tf) and Inverse Document frequency (idf) and used tf.idf as the feature vector representation for the document. Once we were able to compare few questions (~5) against each other, we wrote the complete algorithm with K-means and Cosine-similarity. We also wrote a separate tf.idf script which dumps the computed tf.idf dictionary for any given input file of questions. The K-means script takes the input questions file which was pre-processed with POS tagging, stop words were removed and stemming was applied.

Also using the POS tagged words we identify Adjectives and Verbs in the given input questions file based on the tags using regular expressions and call the Synonym module which replaces the words with their Super Synonym (X-label).

After above processing is done, the input questions file was passed to tf.idf script which generates the tf.idf pickle dump which is loaded by the K-means script to use for cosine-similarity measure. K-means clustering is performed using NLTK cluster module. The number of clusters to be formed was calculated as an empirical analysis on the given input questions file. The resulting cluster object is saved using pickle dump for future classification of Test questions.

The above process was repeated by substituting Euclidean distance as a measure for K-means clustering instead of Cosine-similarity.

4.5 Classification of Test data

Test data comprised of 150 questions belonging to 15 different fields obtained from Stack Overflow using Stack Exchange API. These questions were passed through the same sequence of pre-processing as the questions in training data. Tokenized and POS-tagged questions are then replaced with super synonyms (X-labels) present in the dictionary which was already populated while running questions from training set using Merriam Webster synonym Web API. These test questions are now classified into already existing clusters generated by running K-means algorithm for training data.

NLTK's cluster classify function was used to classify the questions into clusters. The cluster distribution in case of K-means with cosine similarity measure was better when compared to the one with Euclidean measure.

4.6 Sub-clustering

Next step was to find the most relevant question for each test question within the cluster where it belongs. For each test question we performed clustering again by considering the set containing the test question as well as the questions which belonged to the same cluster as that of test question. For this stage of sub clustering we used K-means algorithm with cosine similarity as the measure and tf.idf as the feature vector representation. tf.idf vector for each questions in the main cluster was generated and passed to the K-means algorithm. The new sub-cluster where the test question was clustered into provides most similar or relevant question to the test question.

5 Evaluation

5.1 Evaluation metrics

We added graph generation capability in the code base using the MatPlot library [10] for visual representation of question distribution across clusters.

This helped in easily identifying that Cosine similarity measure was performing better than Euclidean distance (ED), as the distribution was more even in the earlier case.

The training set of 1800 questions was grouped into 60 clusters. As we can see in Figure 1, when using Euclidean distance, one of the clusters had 1200 question within it, whereas when Cosine similarity was used the maximum number of questions in a cluster was only 200.

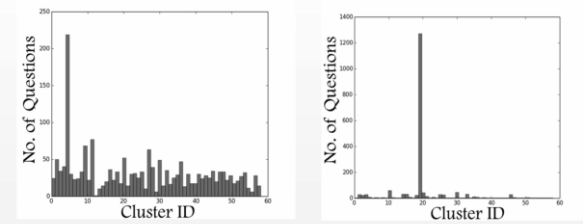

Figure 2: Question distribution over clusters

We then manually annotated a separate set of 150 questions for the testing and divided them into 15 classes. We then checked the result set of the classifier for each of these questions. Correct classification refers to a result set that had questions with same meaning to the input test question. We then calculated precision and recall for all the 15 classes.

The overall F-score was evaluated as the average F-score across all the 15 classes.

Avg. F-score $= (1/n)\sum_{k=1}^{n} F_k$ = 0.61 (for n=15)

5.2 Analysis

The system performs excellent in cases where the questions are from different domains.

For e.g.: A question say, "How to configure ADFS in .NET?" gets correctly grouped into the pool of question related to ADFS and stays away from other groups with questions like "How to reset IIS programmatically in .NET?"

However, in cases of questions that are closely similar with only slight variations, like "How to find minimum in an array in C?" and "How to get minimum in a list?" the system resulted in an overall F-score of 0.61.

5.3 Inference

We understand that cosine similarity works better than ED because the earlier measure takes into consideration the direction (style) of the vector representation as well, whereas the later only considers the magnitude.

Also, representing all synonyms by their respective labels helps in bringing similar questions together. The synonym list generated by Merriam-Webster API is far better than word-net, this had a direct impact on the clustering. So we are hoping that the F-score can be increased a bit by retrieving better synonym list.

Another difficulty that we faced was in generation of POS tagging for the new grammar of user generated query. In the sentence "kth smallest element" all the words are tagged as Nouns. A better

POS tagger will again improve the overall F-score and lead to better clustering of documents.

6 Team Contribution

6.1 Project Contribution

Darshan: Cosine-Similarity standalone script implementation, TF_IDF_Computation script, Wordnik Synonym generator script, Tagged 50 questions manually for test, Designed Poster, Rand Index Evaluation, Created script to aggregate question data & assign unique question IDs, contribution to k-means algorithm script.

Pradeep: K-means Algorithm with Cosine similarity and Euclidean measures script, Tokenizing/POS/Stopword/Stemming tagging script. Collected 400 questions from DataRetrieval Script, Tagged 50 questions manually for test, Super Synonym generator script for X-labels, Sub clustering algorithm for test data, manual F-Score calculation for Test

Vishal: StackOverflow Data Retrieval Script, Collected 1400 questions from DataRetrieval Script, Merriam Webster & Wordnet Synonyms generator Scripts, Tagged 50 questions manually for test, Added Graph plotting for Question distribution, Cluster Purity Evaluation, Sub clustering algorithm for test data, manual F-Score calculation for Test

6.2 Report Contribution

Darshan: 1.Introduction; 4.Techinical Approach 4.1, 4.2, 4.4; 7.References
Pradeep: 2.Related Work; 4.3, 4.5, 4.6 of Technical Approach.
Vishal: 3. Data 5.Evaluation

7 References

[1]http://stackoverflow.com/

[2]https://api.stackexchange.com/

[3]http://www-scf.usc.edu/~vishaljs/nlp/GetQuestions.html

[4]http://www.nltk.org/book/ch05.html

[5]http://snowball.tartarus.org/algorithms/english/stemmer.html

[6]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[7]http://www.dictionaryapi.com/products/api-collegiate-thesaurus.html

[8] http://www.nltk.org/howto/wordnet.html

[9] https://github.com/wordnik/wordnik-python

[10] http://matplotlib.org/devdocs/contents.html