A REPORT ON

RANKING BASED QUESTION ANSWERING SYSTEM

SUBMITTED IN PARTIAL FULFILMENT OF THE COURSE
CS F266 - STUDY ORIENTED PROJECT

BY
VISHAL MITTAL
2017A7PS0080P

TO
DR. YASHVARDHAN SHARMA
Associate Professor,
Computer Science and Information Systems Department
BITS-Pilani, Pilani Campus

DATED: 28th October 2019

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Introduction

The problem of making a fully functional question answering system is one problem which has been very popular among researchers. Querying information from data has now become an essential task. There is a lot of textual data including newspapers, articles, FAQ, documentation, customer service requests, etc. Question Answering (QA) system is an automated computer system that answers questioned posed by humans in natural language. Modern QA systems are of two types: open domain and closed domain. Open-domain QA systems can answer any question asked in human recognizable language while closed domain QA systems are restricted to context. Question Answering systems are widely being used in personal assistants, chatbots and knowledge encyclopedia.

Background

In recent years, with the onset of Deep Learning, there have been enormous advances in the field of question answering. This is because question systems are very useful as they allow users to enter a query based on some facts or stories and the system tries to use the context in the supporting facts and stories to answer the questions effectively instead of just giving out the best-suited keywords. Besides, most of the problems in AI can be modeled as question answering problems and we aim to design a system. Some Question Answering systems available on the web are Quora, StackOverflow, Math.StackExchange, Yahoo Answers, etc. Users post questions on these websites and answers are posted by the community users.

Motivation

Most of the problems in artificial intelligence and Natural language processing can all be modeled as a question answering problems. For example, the task of text summarization can be modeled as a question answering task in the sense that if the user asks the system “What is the summary of the text?”, it can answer him by providing the appropriate summary. Most of the Question Answering Systems are domain-specific. Hence the user is not able to find the correct and most relevant answers to their questions. Some questions asked by users are even very much context-based related to certain organizations to which only those organizations can answer. For example, a question related to cricket can best be answered by cricket-specific websites like Cricbuzz or ESPNCricInfo and not by established Question Answering models.

Objective

In this project, various available Ranking Based Question Answering systems are reviewed and a technique is proposed which selects the best answer from the available QA models using cosine similarity and NLP, and also answers some domain-specific questions which can't be answered by the above systems. A chrome-extension is presented to present the above approach. A comparative study between cosine similarity and Euclidian distance method is also shown. Finally, a question classification model is built which classifies the questions into one of the 10 categories based on its context.

Related Work

There are many user based question answering systems available on the web. Users post questions to these QAs and these questions are answered by other users who might know answers to these questions. Some of the very famous QA models are Quora, Stack Overflow, Stack Exchange, Yahoo answers, Fluther, etc. These QA systems are very specific to the type of questions asked. For example, Quora specializes in answering subjective or open-ended questions that can belong to various domains, whereas Stack Overflow has a speciality in providing answers to technical and computer science-related questions. Similar to these, there are many other domain-specific web applications where users can find answers to certain domain-specific questions. There are also many context-based question answering models that use deep learning approach to find answers to the questions from the given context. These models are trained on organizational data such as the one developed for BITS Pilani because the data about them is available to these organizations only and sometimes not available in public domain.

Proposed Technique and Algorithm

The architecture of the system comprises of 4 modules:

1. Chrome Extension
2. Answer Selection Web Service
3. Question Answering System
4. Question Classification Model

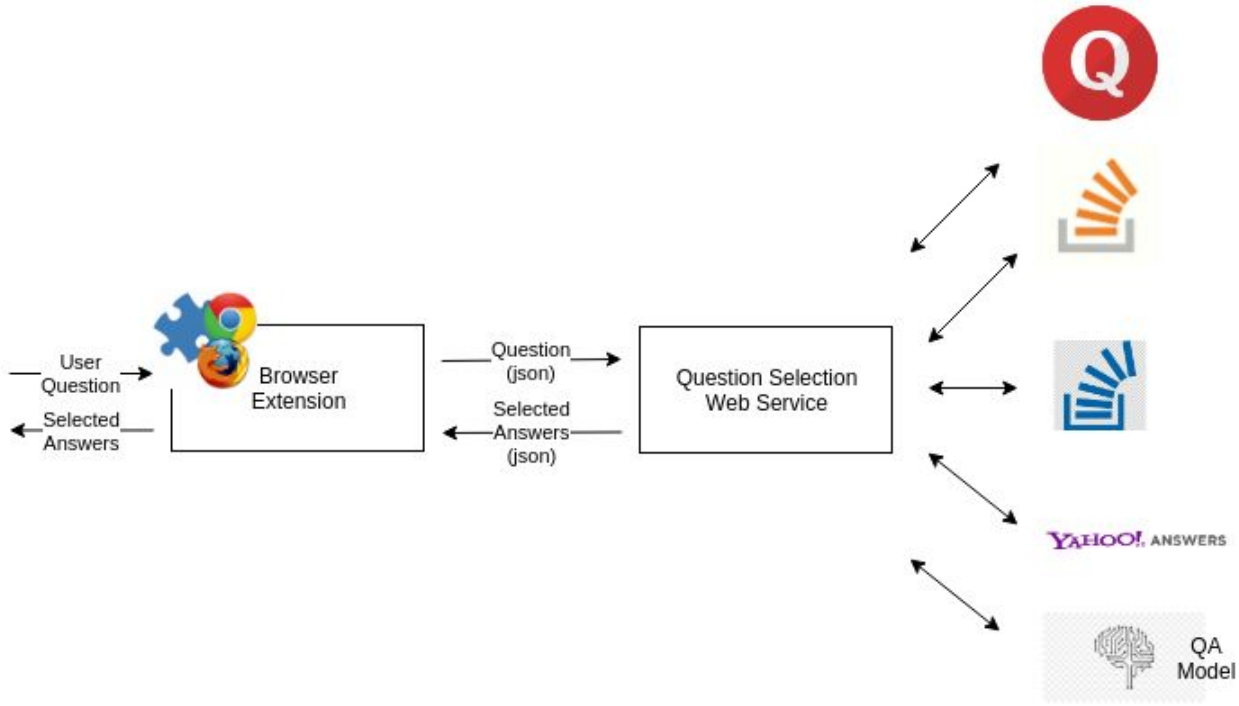

Fig. 1: Workflow of the project

Chrome/Firefox Extension

Extensions are small software programs that are used for customizing the browsing experience. They allow users to efficiently use Chrome functionality and behaviour to individual needs or preferences. They are built on web technologies like HTML, CSS and JavaScript. Extension can be downloaded through the extension marketplace.

Extension consists of a question interface where the user can ask any question. This question is sent to the webserver which is running answer selection web service. The response from the service consists of top 5 relevant questions/answers and their links to corresponding QA systems. The user can navigate through these links to find the answer of his relevance.

Question Selection Web Service

This web service does all the computation for finding the best relevant answer. The question asked by the user is passed to the web service using JSON over the HTTP protocol. The question is then searched on the top QA systems which are available on the web. The QA systems return with the various questions asked by the users which are similar to the asked question. All the returned questions are then matched with the asked question using similarity measurement techniques. Here in our approach, we are cosine similarity as the measurement technique. Links of questions which are found to be most similar to the asked question are returned to the client as

JSON. Some questions might be domain-specific where the question can only be answered by the deep learning pre-trained model. These questions along with selected context are provided to a pre-trained model which provides the answers.

Question Answering System

There are online Question Answering Systems available on the web. The answer selection service uses these QA systems to find the most relevant question to the asked question by the user using cosine similarity measure. The Question Answering System can also be sent to a pre-trained model which provides answer to the user domain/context-specific question using deep learning approach and keyword matching.

Question Classification Model

The machine learning model built using linear SVM model classifies the question into one of these classes - Society, Science, Health, Education, Technology Sports, Finance, Entertainment, Relationships, Politics - based on the context of the question. This helps in finding the answer to the asked question from the best sources by preparing a dedicated list of websites for each class. An additional class 'BITS Pilani' is defined to direct users to the BITS Pilani QA Model for specific queries regarding BITS.

Implementation Details

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing.

1. Tokenization

It means breaking up a sentence into individual tokens. An example of how one might actually tokenize something into tokens with the NLTK module:

```
from nltk.tokenize import word_tokenize

EXAMPLE_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great,
and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."
 print(sent_tokenize(EXAMPLE_TEXT))
```

2. Stop words removal

The idea of NLP is to do some form of analysis where the machine can understand what the text implies. One of the most important forms of pre-processing is filtering out useless data. In NLP, useless words or data are also referred to as stop words since some words carry more meaning than other words. Stop words are words that just contain little meaning and should be removed for better analysis.

```
from nltk.corpus import stopwords
```

Here is the list:

```
set(stopwords.words('english'))
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during',
'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such',
'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'fr
>>>    om', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his',
'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our',
'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any',
'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that',
'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself',
'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't',
'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}
```

3. Lemmatizing with NLTK

Lemmatization involves removal of inflectional endings and to return the base or dictionary form of a word, which is known as the *lemma.* Stemming may cause the removal of derivational affixes. Hence, lemmatization is considered a better technique than stemming as it works on vocabulary and morphological analysis of words.

4. Use of Sklearn

Scikit-learn is a free software ML library for the Python. It features various clustering, regression and classification algorithms including random forests, k-means, support vector machines, gradient boosting and DBSCAN, and is designed to operate with the Python numerical library Numpy and scientific library SciPy.

```
from sklearn.feature_extraction.text import CountVectorizer
```

5. Measuring the Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle in the interval [0,2π). It is thus a judgment of orientation and not magnitude It is particularly used in positive space, where the outcome is neatly bounded in [0,1]. Unit vectors are maximally "similar" if they're parallel and maximally "dissimilar" if they're orthogonal (perpendicular). This is analogous to the cosine, which is unity (maximum value) when the segments subtend a zero angle and zero (uncorrelated) when the segments are perpendicular. These bounds apply for any number of dimensions, and cosine similarity is most commonly used in high-dimensional positive spaces. For example, in text mining and information retrieval, each term is assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter. The technique is also used to measure cohesion within clusters in the field of Data Mining.

Cosine distance is a term often used for the complement in positive space, that is:
Dc(A, B) = 1 - Sc(A, B)
where Dc is the cosine distance and Sc is the cosine similarity.

Cosine similarity is popular because it is very efficient to evaluate, especially for sparse vectors, as only the non-zero dimensions need to be considered.
The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:
$a.b=||a||\ ||b||\ \cos(\theta)$

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude as

Similarity = cos(θ) = A.B/||A|| ||B|| = sum(AiBi) from 1 to n/(sum(Ai) *sum(Bi)) from 1 to n

where Ai and Bi are components of vector A and B respectively.

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality (decorrelation), and in-between values indicating intermediate similarity or dissimilarity. For text matching, the attribute vectors A and B are usually the term frequency vectors of the documents. The cosine similarity can be seen as a method of normalizing document length during comparison.

Cosine Similarity will generate a metric that tells how related the two documents are by looking at the angle instead of magnitude, similar to the examples below:

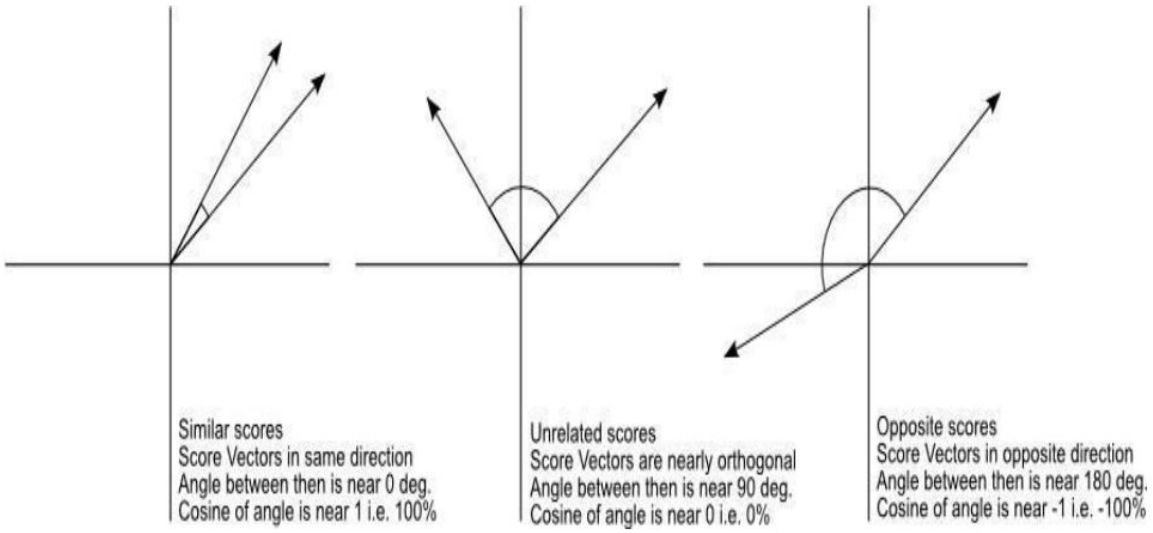

Fig. 2: The Cosine Similarity values for different documents, 1 (same direction), 0 (90 deg.), -1 (opposite directions).

Even if we had a vector pointing to a point far from another vector, they still could have a small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents.

Now we have a Vector Space Model of documents modelled as vectors (with TF-IDF counts) and also have a formula to calculate the similarity between different documents in this space.

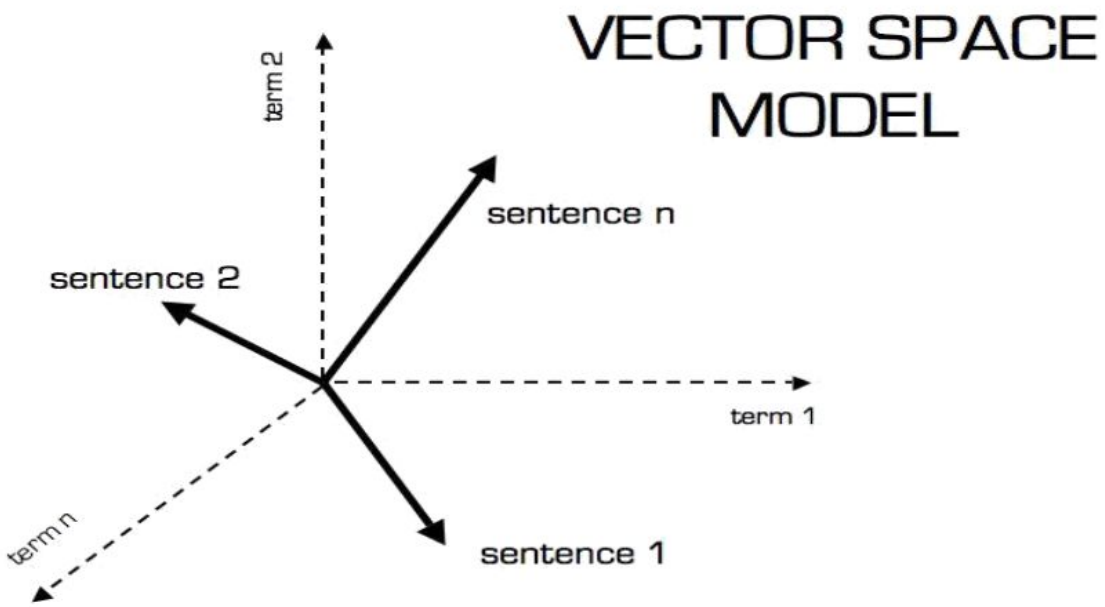

Fig. 3: Vector Space Model

Euclidean Distance

Euclidean distance is the most commonly used distance to measure text similarity. When data is dense or continuous, this is the best proximity measure.

The Euclidean distance between two points is the length of the route connecting them. The Pythagorean theorem gives this distance between two points.

The Euclidean distance between points p and q is the length of the line segment connecting them (PQ). In Cartesian coordinates, if p = (p1, p2,..., pn) and q = (q1, q2,..., qn) are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

DIFFERENCE BETWEEN COSINE SIMILARITY AND EUCLIDEAN DISTANCE SIMILARITY

EUCLIDEAN DISTANCE SIMILARITY

Euclidean Similarity calculates the distance between two users and then it tries to find out the similarity. This makes sense if you think of users as points when there are many dimensions (as many dimensions as the items), whose coordinates are preference values. This similarity metric calculates the Euclidean Distance (d) between two such user points. If you look at User 1, the distance is calculated as 0, because for this particular user the distance is 0. Similarity will be calculated using the formula, 1/1+d, where d is the distance.

COSINE SIMILARITY

Cosine similarity metric finds the normalized dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any angle other than 0°.
Hence, it is a judgment of orientation and not magnitude. Two vectors with the same orientation have a cosine similarity of 1 while two vectors at 90° have a similarity of 0, and two vectors diametrically opposite to each other have a similarity of -1, independent of their magnitude. Cosine similarity is often used in positive space, where the outcome is uniformly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors.

MODEL FOR QUESTION CLASSIFICATION

Model: Linear Support Vector Machine Classifier

Approach:

1. Text Exploration
2. Text Cleaning
3. Obtaining POS Tags, Identifying Named Entities, Lemmas, Syntactic Dependency Relations and Orthographic Features.
4. Using the obtained properties as Features.
5. Using a Linear SVM model on the engineered features.

Features used: Named Entity Recognition + Lemmas + POS Tags

Dataset

We used Yahoo! Answers QA Topic Classification Dataset. The corpus contains 4483032 questions and their answers.

DESCRIPTION

The Yahoo! Answers topic classification dataset is a human labelled dataset constructed using 10 largest main categories. Each class contains 140,000 training samples and 6,000 testing samples. Therefore, the total number of training samples is 1,400,000 and testing samples 60,000 in this dataset. From all the answers and other meta-information, only the best answer content and the main category information were used.

Classes for classification:

1. Society & Culture - e.g. what are the social views of people from different cultures on abstinence?
2. Science & Mathematics - e.g. What is fringe in optics?
3. Health - e.g. What the hell is restless leg syndrome?
4. Education & Reference - e.g. what is the past tense of tell?
5. Computers & Internet - e.g. Do You have the code for the Luxor game?
6. Sports - e.g. Who is going to win the FIFA World CUP 2006 in Germany?
7. Business & Finance - e.g. What is secretary as corporation director?
8. Entertainment & Music - e.g. where can I download mp3 songs?
9. Family & Relationships - e.g. who's ready for Christmas?
10. Politics & Government - e.g. Isn't civil war and oxymoron?

Experiments and Results

Extension was developed for Google Chrome and the web service was deployed on the Flask server. The implemented Question Answering System is demonstrated in the following figures below.

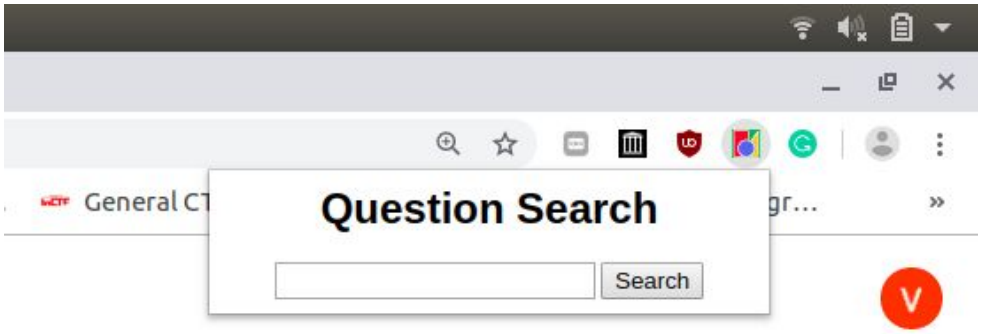

Fig.4: Chrome Extension for Search Wizard

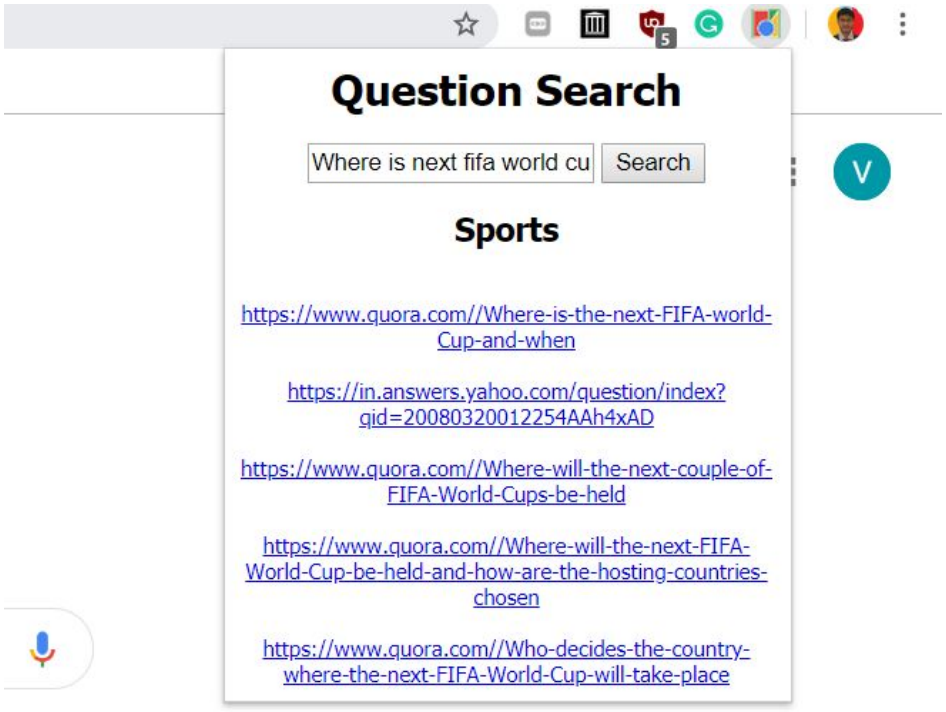

Fig.5: Example Question Search

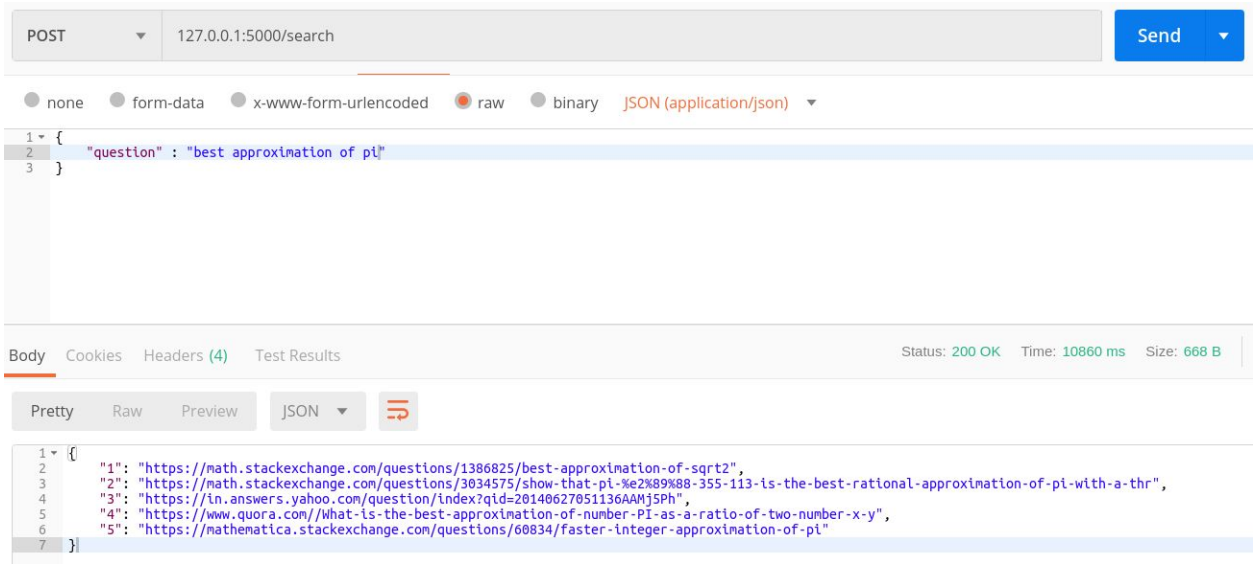

Fig.6: Web Service

From the figures, it can be observed that the most relevant answer to the question related to mathematics was found on math stack exchange. Figure 5 shows the interface of the web service which takes a question as input over HTTP and returns the best relevant response to the asked question along with the category of the question.

Accuracy of linear SVM model turned out to be **66.316%**

Conclusion and Future Work

In the presented solution, Question selection based Question Answering system was implemented. The complete system was divided into four modules namely Extension, Question selection Web Service, QA system and Question Classification Model. The extension provided the user interface whereas the Question Selection Web services provided the most relevant question matching to user query using the pre-built QA systems. The whole system was tested with various user queries. Currently, we are using cosine similarity for measuring the similarity between the questions. For future work, QA system accuracy will be improved by using deep learning approach for measuring similarity between questions. One of the proposed technique that can be used is Siamese Manhattan LSTM which will be applied later to get better results.

Work Update

Work Completed

1. Developed Chrome Web Extension
2. Developed Question Selection Web Service
3. Selected most relevant Questions from QA systems using the developed web service.
4. Question similarity matching using cosine similarity
5. Applied Classification techniques to find the domain of the asked question

Work Remaining

1. Deploying and Integrating Context-based QA model with the above-developed system
2. Applying deep learning techniques such as Siamese Manhattan LSTM for similarity matching of questions

References

1. Liviu P. Dinu, Radu Tudor Ionescu “A Rank-Based Approach of Cosine Similarity with Applications in Automatic Classification”
2. Huang, Anna. "Similarity measures for text document clustering." *Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*. Vol. 4. 2008.
3. Mueller, Jonas, and Aditya Thyagarajan. "Siamese recurrent architectures for learning sentence similarity." *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.
4. Olalere Williams “High-Performance Question Classification Using Semantic Features” Stanford University, CS224N Spring 2010
5. Harish Tayyar Madabushi, Mark Lee “High Accuracy Rule-based Question Classification using Question Syntax and Semantics” The 26th International Conference on Computational Linguistics, 2016.
6. Zhiheng Huang, Marcus Thint, Zengchang Qin, “Question Classification using Head Words and their Hypernyms” Conference on Empirical Methods in Natural Language Processing 2008
7. Li Xin, HUANG Xuan-Jing, WU Li-de “Question Classification using Multiple Classifiers”