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: 19th 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 structured and unstructured data has become very important. There is a lot of textual data, FAQ, newspapers, articles, documentation, user cases, 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.

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 eg. Quora specializes in answering subjective or open-ended questions that can belong to various domains, whereas Stack Overflow has a specialty 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 3 modules:

1. Chrome/Firefox Extension
2. Answer Selection Web Service
3. Question Answering System

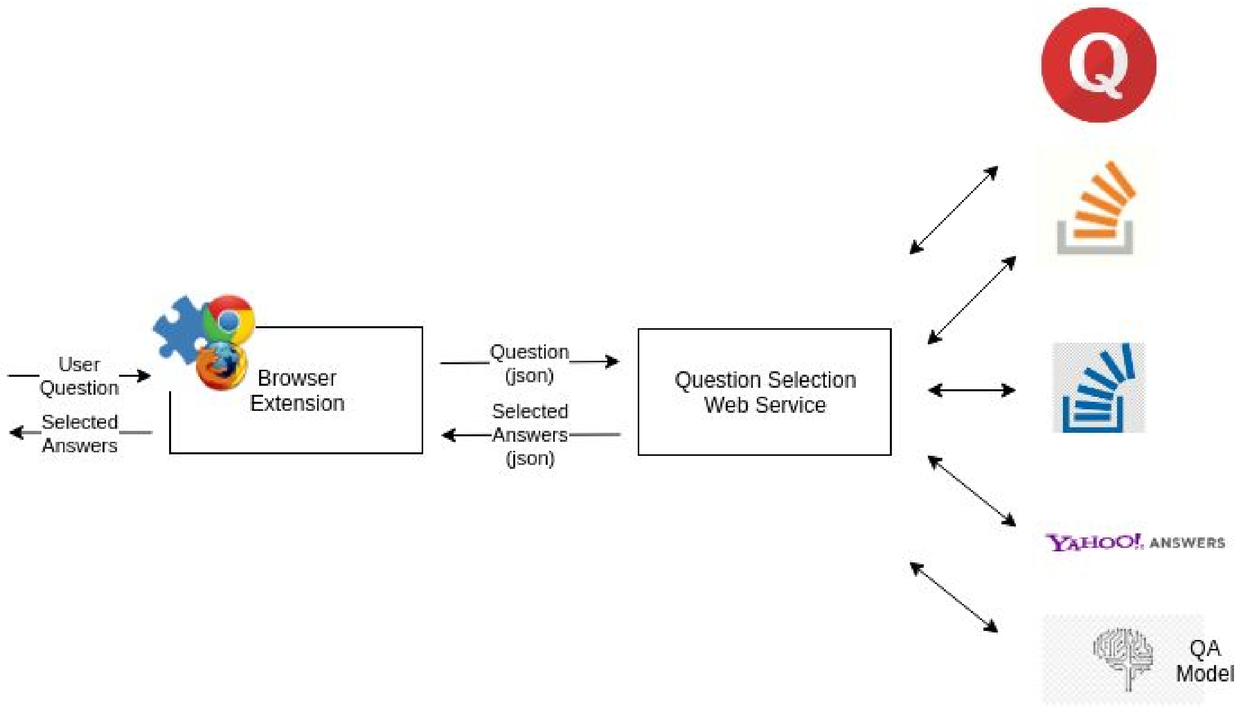


Fig. 1: Workflow of the project

Chrome/Firefox Extension

Extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on web technologies such as HTML, JavaScript, and CSS. 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

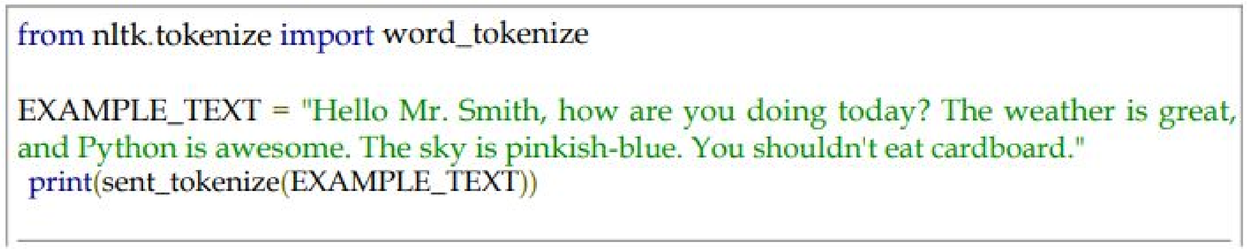
There are online Question Answering Systems available on the web. The answer selection service uses these QA systems to find the most relevant question to the asked question by the user using cosine similarity measure. The Question Answering System can also be sent to a pre-trained model which provides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

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

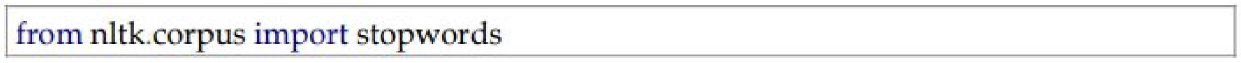
These are the words we most commonly hear upon entering the Natural Language Processing (NLP) space. An example of how one might actually tokenize something into tokens with the NLTK module:



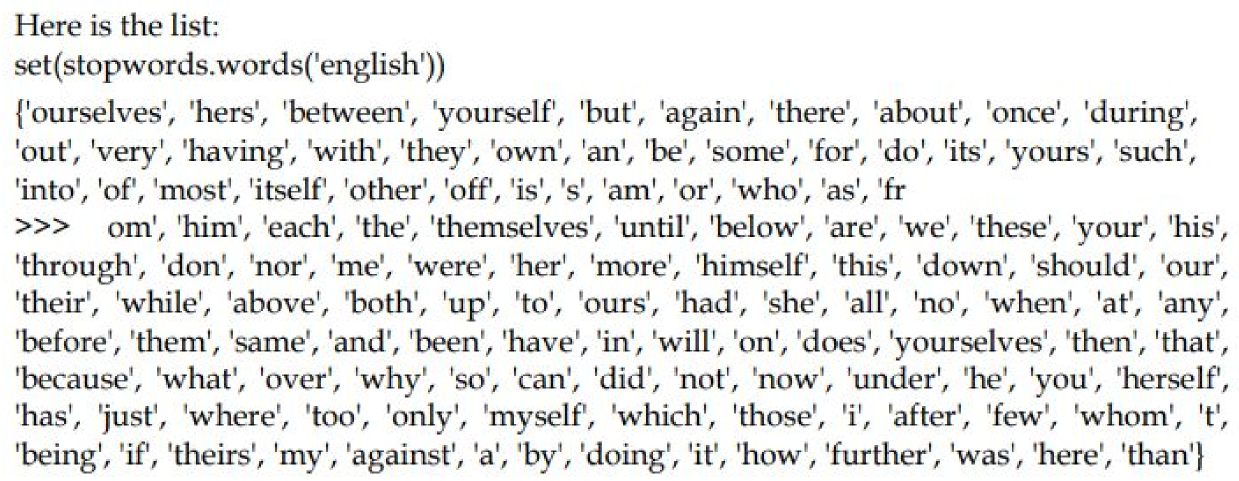
```
from nltk.tokenize import word_tokenize

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 Natural Language Processing is to do some form of analysis, or processing, where the machine can understand, at least to some level, what the text means, says or implies. This is an obviously massive challenge, but there are steps to doing it that anyone can follow. The process of converting data to something a computer can understand is referred to as "pre-processing." One of the major forms of pre-processing is going to be filtering out useless data. In natural language processing, useless words (data), are referred to as stop words. Some words carry more meaning than other words. Stop words are words that just contain no meaning, and we want to remove them.



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



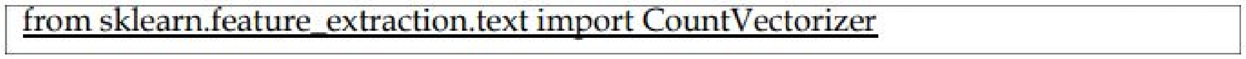
```
Here is the list:
set(stopwords.words('english'))
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during',
'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such',
'into', 'of', 'most', 'itself', 'other', 'off', '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 machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and 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 $[0,2\pi)$. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity 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 information retrieval and text mining, each term is notionally 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, $\cos(\theta)$, is represented using a dot product and magnitude as

Similarity = $\cos(\theta) = A.B/||A||\ ||B||$ = 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 says how related are two documents by looking at the angle instead of magnitude, like in the examples below:

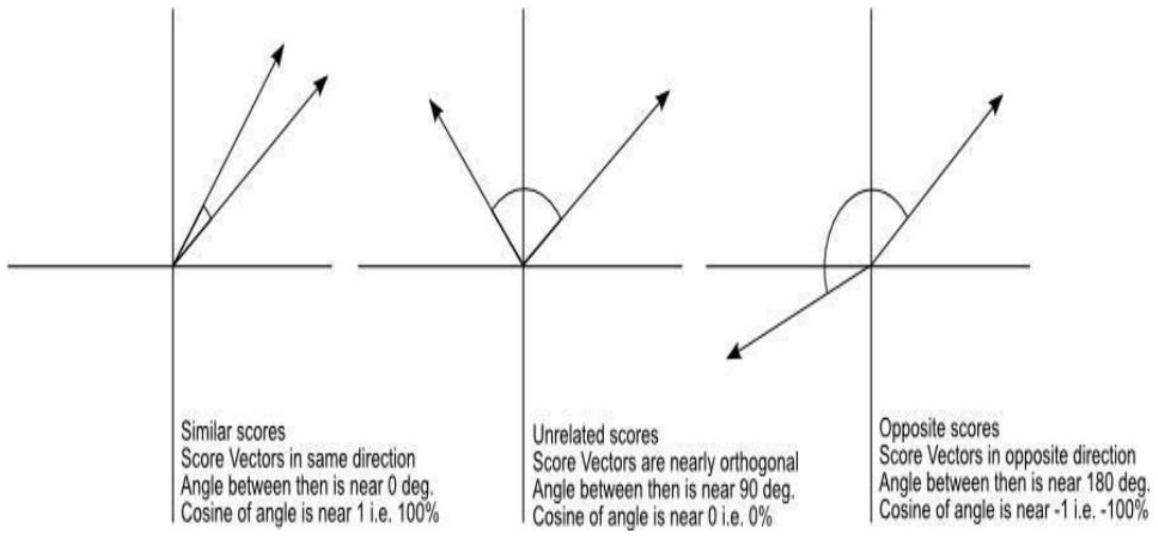

Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

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 modeled as vectors (with TF-IDF counts) and also have a formula to calculate the similarity between different documents in this space.

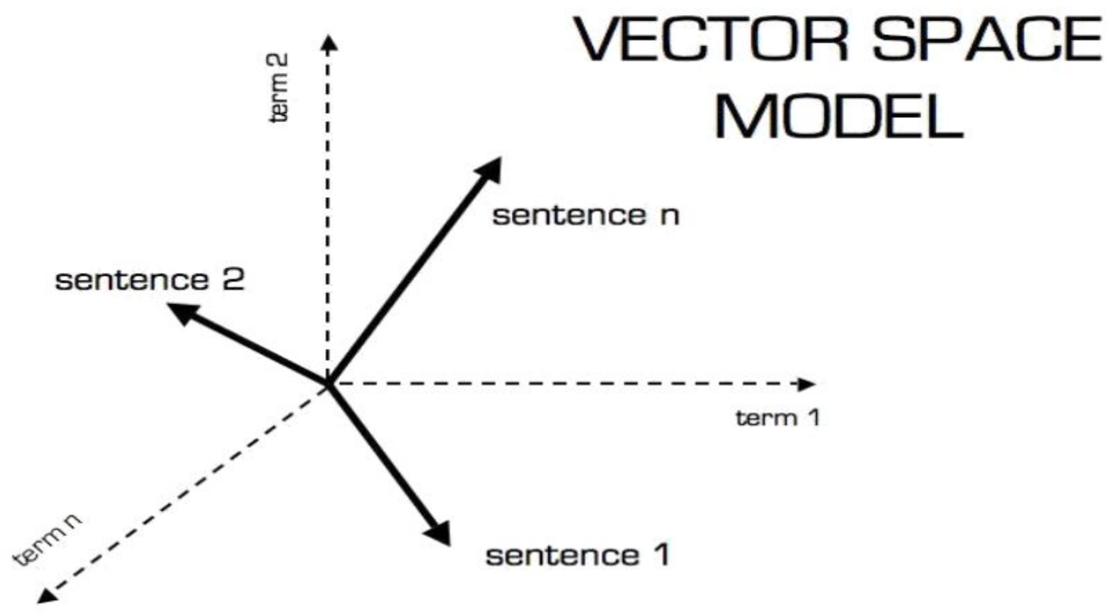


Fig. 3: Vector Space Model

Euclidean Distance

Euclidean distance is the most common use of distance. When data is dense or continuous, this is the best proximity measure.

The Euclidean distance between two points is the length of the path 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 other angle.
It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.
Cosine similarity is particularly used in positive space, where the outcome is neatly 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.

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.

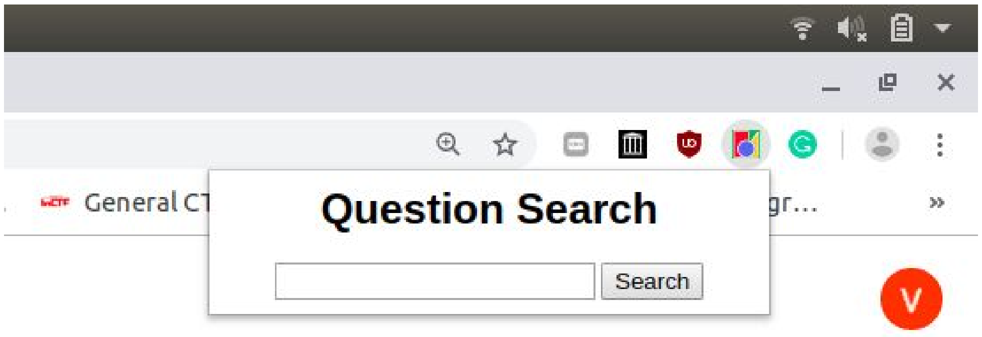
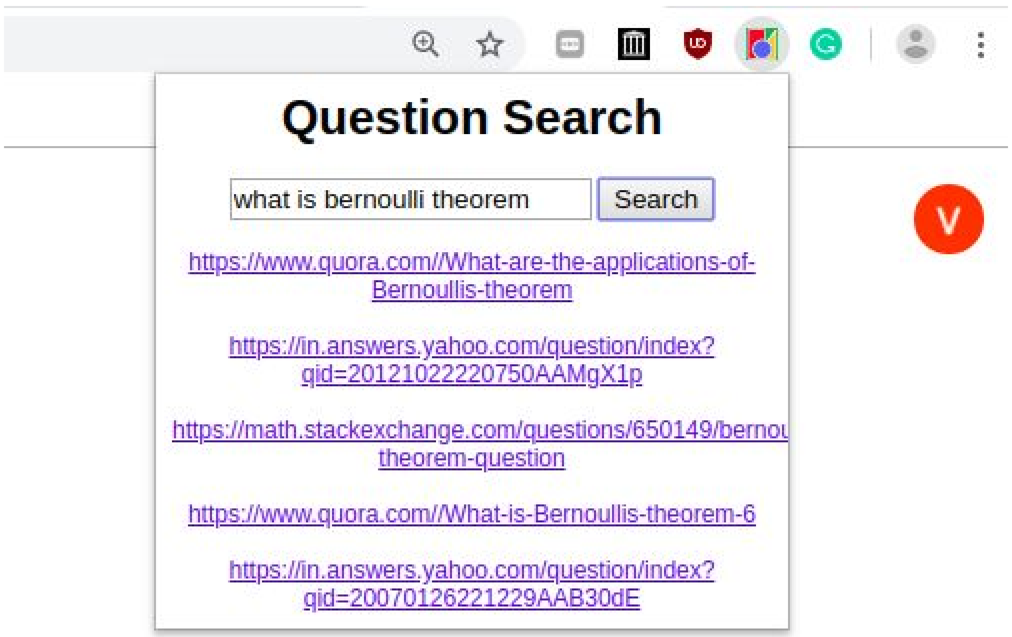


Fig.4: Chrome Extension for Search Wizard


Question Search

what is bernoulli theorem Search

https://www.quora.com//What-are-the-applications-of-Bernoullis-theorem

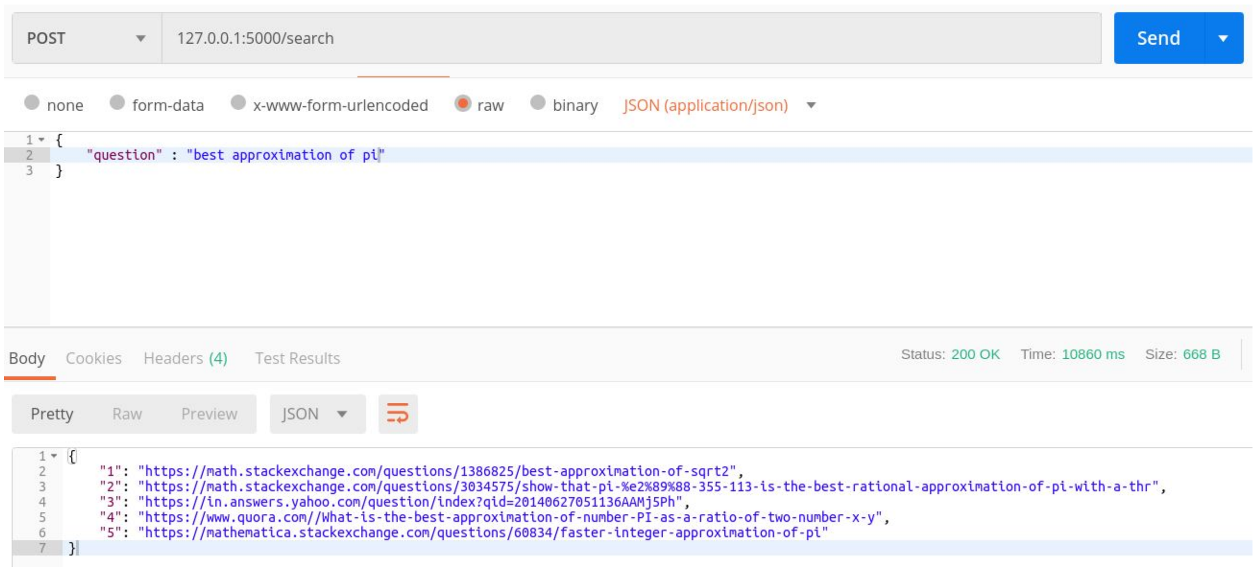
https://in.answers.yahoo.com/question/index?qid=20121022220750AAMgX1p

https://math.stackexchange.com/questions/650149/bernou theorem-question

https://www.quora.com//What-is-Bernoullis-theorem-6

https://in.answers.yahoo.com/question/index?qid=20070126221229AAB30dE

Fig.5: Example Question Search



POST 127.0.0.1:5000/search

raw — JSON (application/json)

```
{
    "question" : "best approximation of pi"
}
```

Status: 200 OK Time: 10860 ms Size: 668 B

```
{
    "1": "https://math.stackexchange.com/questions/1386825/best-approximation-of-sqrt2",
    "2": "https://math.stackexchange.com/questions/3034575/show-that-pi-%e2%89%88-355-113-is-the-best-rational-approximation-of-pi-with-a-thr",
    "3": "https://in.answers.yahoo.com/question/index?qid=20140627051136AAMj5Ph",
    "4": "https://www.quora.com//What-is-the-best-approximation-of-number-PI-as-a-ratio-of-two-number-x-y",
    "5": "https://mathematica.stackexchange.com/questions/60834/faster-integer-approximation-of-pi"
}
```

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.

Conclusion and Future Work

In the presented solution, Question selection based Question Answering system was implemented. The complete system was divided into three modules namely Extension, Question selection Web Service and QA system. 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 techniques that can be used is Siamese Manhattan LSTM which will be applied later to get better results. Classification techniques will also be applied for domain selection of questions.

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

Work Remaining

1. Deploying and Integrating Context-based QA model with the above-developed system
2. Applying Classification techniques to find the domain of the asked question
3. 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.