CS685A: Data Mining
Project Report

GROUP-23

AMEYA KARANJKAR 20111011(ameya20@iitk.ac.in)
TUSHAR GAUTAM 20111071(tushark20@iitk.ac.in)
ABHAS KUMAR 20111001(ababhas20@iitk.ac.in)
JAY VORA 20111023(jvora20@iitk.ac.in)
ANUP ROY 20111403(puna20@iitk.ac.in)

Title : Similar Question Ranking from Q&A sites

Supervised By: Dr. Arnab Bhattacharya

6th December 2020

Contents

1 ABSTRACT

Community Question Answering (CQA) has become a primary means for people to acquire knowledge, where people are free to ask questions or submit answers. To enhance the efficiency of the service, similar question identification becomes a core task in CQA which aims to find a similar question from the archived repository whenever a new question is asked. However, it has long been a challenge to properly measure the similarity between two questions due to the inherent variation of natural language, i.e., there could be different ways to ask a same question or different questions sharing similar expressions.

Most existing methods measure the similarity between questions based on the Semantic. We need to compute the similarity in meaning between two texts. This type of Text similarity is often Computed by first Embedding the two texts and then computing the Cosine Similarity between Them. In this Project We Worked On "Pre-trained encoders" as Our Similarity Methods. Pre-trained sentence encoders Typically they have been trained on a range of supervised and unsupervised tasks, in order to capture as much universal semantic information as possible. Several such encoders are available. We'll take the Google Sentence Encoder into Account also Experiments carried out on a real world dataset Scrapped from Stackoverflow, Stackexchange, Gateoverflow and Superuser.

2 PROBLEM STATEMENT

To Find Semantically Similar Questions to a given Question and Rank them.

2.1 Problem Definition

The problem is defined as follows: Given a question q, identify questions in the corpus that are similar and related to this Question semantically.

Take an example of q being, "How can I get rid of my fear of swimming?" We show questions which are "related" to this question – one of the criteria is semantic identity or exact question match. Hence, It would contain questions that are exact duplicates of the above question and also paraphrases of the form

"How can I overcome my fear of swimming?"
In our definition of paraphrasing, there needs to be at least one semantic identity concept between two questions qualified as paraphrases of each other.

3 SOURCE OF DATASETS

We have used Community-based Question Answering (CQA) sites, Such communities have built up huge archives of question-answer pairs that are continuously accumulating questions. These platforms fulfill our requirement of relevant data needed for this project.

Source	**Description**
StackExchange	Network of various Q&A websites
StackOverflow	Features Q&A on topics in computer programming.
GateOverflow	Q&A on GATE Question
SuperUser	Q&A site for computer enthusiasts.

Table 1: Showing Sources of Datasets we have used.

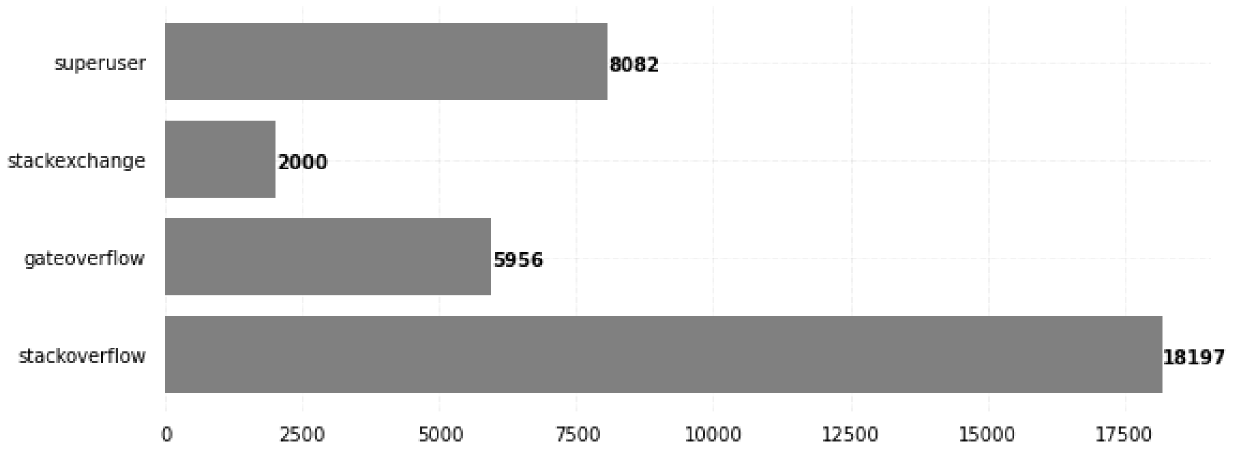


Figure 1: Number of Training examples from each of the source

4 INTRODUCTION AND MOTIVATION

Community-based Question Answering(CQA) sites provides platform for users to interact and respond to other users questions or post their own questions for other users to answer, StackOverflow, StackExchange are some of the examples of such online QA services. These services have created a huge repository of human generated question and answers. There's a lot of research happening in treating this "user generated content" differently than most documents on the web are treated. This leads to a different set of search, browse and presentation problems for this "web" of user generated content. Different models for querying, ranking of results, are evolving for this form of data present in QA form.

One of the features that most of these services wish to offer is to display similar and related questions for a particular "query" question. Unlike a normal text document, a question and answer page is more structured where the most useful portion describing the intent of the page is the question. Traditional document similarity measures use bag of words kind of models and approximate similarity between two documents to mean high overlap of the terms in those documents. However, in order to determine similarity between two questions, it becomes important to define a similarity measure that is targeted more at the sentence level and which captures the meaning and context of the sentence.

In this project, along with the collection of quality data from multiple sources and developing techniques for measuring similarity of two questions, we have also looked at standard pre-trained encoders which convert the text sentences into numerical vector representations (embeddings) on which various similarity measures can be used. The main aim of the project is however to satisfactorily predict and output some set of similar questions given a user query.

5 DATA PREPARATION

5.1 Data Extraction

We have used web scraping to access a website's content(mentioned in Table 1) and create our raw dataset. Python modules and Libraries used to pull data from web pages and locate the required content within HTML structure of response we got from cQA sites:
1.Requests
2.Beautiful Soup 4
3.Selenium
4.Scrapy

In order to populate the dataset, we crawled over questions of above mentioned cQA sites (Table 1). The questions were of varying lengths, of different structures and belonging to various categories e.g. Computers network, maths, programming, Administration, Gate Question, systems, Data Science etc. We extracted required data by just showing scraper the correct HTML elements. Each **HTML** tag refers to a particular element, we used tags like <**p**> for paragraphs; <**a**> or anchor tag for hyperlinks; <**h1**>, <**h2**>, etc. for text headers; <**div**> for dividers, <**tr**> for table rows, and <**td**> for table columns for extraction of required fields.

5.2 Data Characteristics

In total, we acquired around **34,235** examples for our training dataset from above mentioned sources. Web scrapping followed by extraction of relevant contents of HTML responses we put data into CSV format(different CSV file for each cQA site). Initially **18** features for each question were extracted, brief description of some of these features follows next.

- **ID**: Unique number assigned to each question. (Type: Number)
- **Title**: Title of the question. (Type: String)

- **Question**: Actual content of the Question. (Type: String)
- **Views**: Total number of views of users on the question. (Type: Number)
- **Votes**: Total number of votes by users on the question. (Type: Number)
- **Bookmarked**: Total number of bookmarks on the question. (Type: Number)
- **Tags**: Various Tags associated to the question. (Type: String)
- **Link**: Full link of the question. (Type: String)
- **Answered**: Number of accepted answers to the question. (Type: Number)
- **Answer**: All Accepted Answers to the question. (Type: String)
- **AnswerVotes**: Total number of Up votes on all answers of the question. (Type: Number)
- **SuggestedAnswers**: All suggested answers to the question. (Type: String)
- **SuggestedAnswersVotes**: Total number of Up votes on Suggested answers to the question. (Type: Number)
- **Source**: Source Website of the Question. (Type: String)

5.3 Pre-Processing

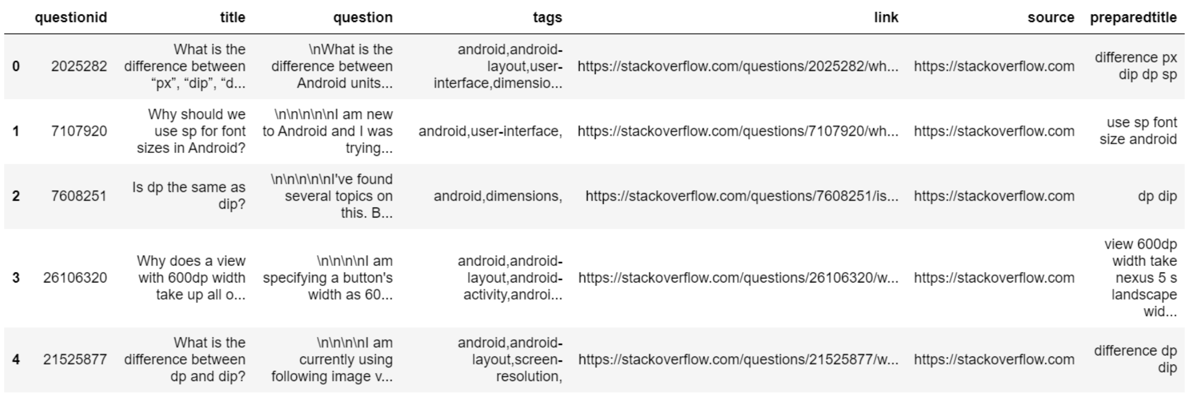
5.3.1 Feature Selection.

From raw Feature Set, we selected a subset of feature that looked more relevant(no feature engineering techniques have been used till now , this has been done manually but we do intend to use such techniques to further extract good features) for the task of finding similar questions(Semantically) and ranking them on the basis of properties of their answers.

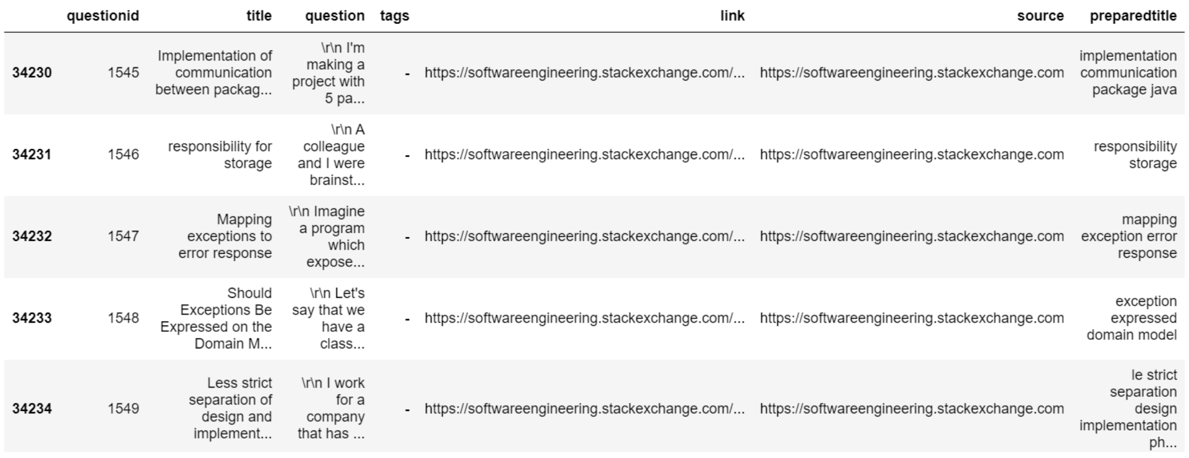
5.3.2 Text Based Pre-processing.

1. **Duplicate** Questions were removed.
2. Questions not having both accepted and suggested answer were removed.
3. All characters were converted to **Lowercase**.
4. White spaces were removed.
5. Punctuation like **(, . ?...,***,')** and accent marks were removed.
6. Stop words of English language **the,is,are** etc were removed.
7. Non-**ASCII** characters like **Ḡ,Ÿ** were converted to ASCII.
8. **Lemmatization**- Each word was reduced to its root word like **feet->foot** (morphological variants).

**Finally we prepared our dataset(in CSV) after merging all processed raw data files.



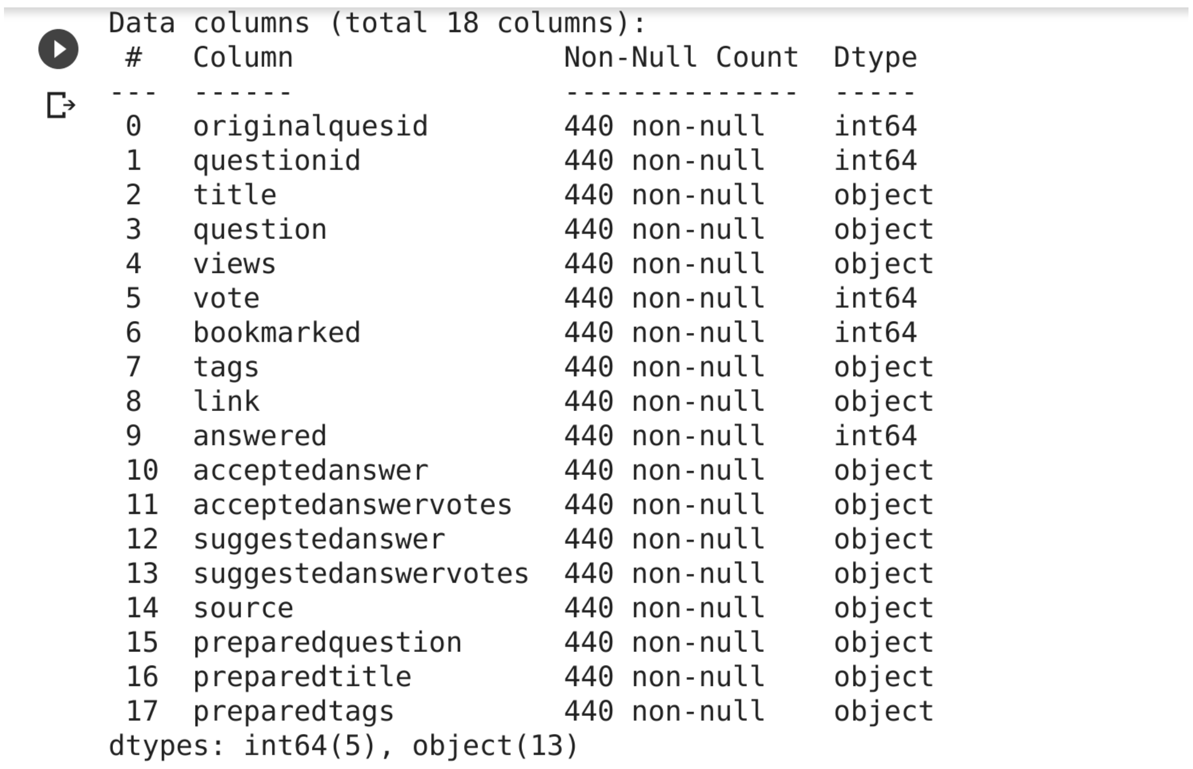
	questionid	title	question	tags	link	source	preparedtitle
0	2025282	What is the difference between "px", "dip", "d...	\nWhat is the difference between Android units...	android,android-layout,user-interface,dimensio...	https://stackoverflow.com/questions/2025282/wh...	https://stackoverflow.com	difference px dip dp sp
1	7107920	Why should we use sp for font sizes in Android?	\n\n\n\n\nI am new to Android and I was trying...	android,user-interface,	https://stackoverflow.com/questions/7107920/wh...	https://stackoverflow.com	use sp font size android
2	7608251	Is dp the same as dip?	\n\n\n\n\nI've found several topics on this. B...	android,dimensions,	https://stackoverflow.com/questions/7608251/is...	https://stackoverflow.com	dp dip
3	26106320	Why does a view with 600dp width take up all o...	\n\n\n\nI am specifying a button's width as 60...	android,android-layout,android-activity,androi...	https://stackoverflow.com/questions/26106320/w...	https://stackoverflow.com	view 600dp width take nexus 5 s landscape wid...
4	21525877	What is the difference between dp and dip?	\n\n\n\nI am currently using following image v...	android,android-layout,screen-resolution,	https://stackoverflow.com/questions/21525877/w...	https://stackoverflow.com	difference dp dip



	questionid	title	question	tags	link	source	preparedtitle
34230	1545	Implementation of communication between packag...	\r\n I'm making a project with 5 pa...	-	https://softwareengineering.stackexchange.com/...	https://softwareengineering.stackexchange.com	implementation communication package java
34231	1546	responsibility for storage	\r\n A colleague and I were brainst...	-	https://softwareengineering.stackexchange.com/...	https://softwareengineering.stackexchange.com	responsibility storage
34232	1547	Mapping exceptions to error response	\r\n Imagine a program which expose...	-	https://softwareengineering.stackexchange.com/...	https://softwareengineering.stackexchange.com	mapping exception error response
34233	1548	Should Exceptions Be Expressed on the Domain M...	\r\n Let's say that we have a class...	-	https://softwareengineering.stackexchange.com/...	https://softwareengineering.stackexchange.com	exception expressed domain model
34234	1549	Less strict separation of design and implement...	\r\n I work for a company that has ...	-	https://softwareengineering.stackexchange.com/...	https://softwareengineering.stackexchange.com	le strict separation design implementation ph...

5.4 Test DataSet Preparation

We need to have some ground truth, known result, in order to evaluate our model.
The Q/A sites that we have considered also have one feature of related/ duplicate questions on some question pages. We have used this feature to get the related/ duplicate questions for a question, on the assumption that related questions are likely to be similar.
We have taken '440' such questions from our corpus and for these questions we have used the above feature to get the related/similar questions and paired them up.
These '440' questions have been removed from the corpus as they are now a part of the test data-set. We then checked these questions manually, if the related questions mapped in the above step are actually similar or not. If not then we found similar questions manually and paired them up with such questions. This way the test corpus was prepared where each question has one or two similar questions paired with it.
We now run our model over each of these test questions and find the top 'n' similar questions using our model. We then assessed the performance of our model based on whether at least one of the already known similar questions appeared in the list of similar questions predicted by our model. An example of the test dataset columns is given in the below figure.



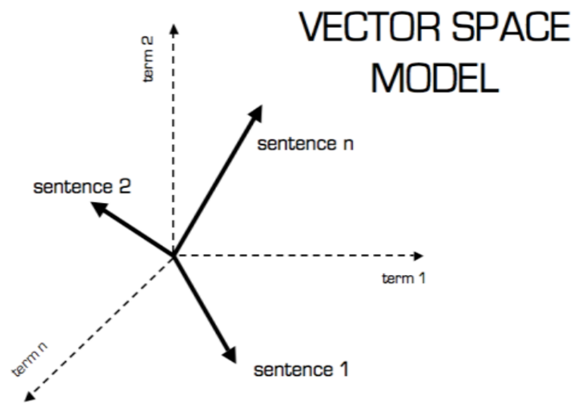
```
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   originalquesid        440 non-null    int64 
 1   questionid            440 non-null    int64 
 2   title                 440 non-null    object
 3   question              440 non-null    object
 4   views                 440 non-null    object
 5   vote                  440 non-null    int64 
 6   bookmarked            440 non-null    int64 
 7   tags                  440 non-null    object
 8   link                  440 non-null    object
 9   answered              440 non-null    int64 
 10  acceptedanswer        440 non-null    object
 11  acceptedanswervotes   440 non-null    object
 12  suggestedanswer       440 non-null    object
 13  suggestedanswervotes  440 non-null    object
 14  source                440 non-null    object
 15  preparedquestion      440 non-null    object
 16  preparedtitle         440 non-null    object
 17  preparedtags          440 non-null    object
dtypes: int64(5), object(13)
```

Test Dataset: Used to provide an unbiased evaluation of Our final model.

6 BACKGROUND KNOWLEDGE

6.1 Vector space model

We have a Vector Space Model of sentences modeled as vectors with respect to the terms and also have a formula to calculate the similarity between different pair of sentences in this space.



Vector space model is a model in which each text document is represented as a numerical vector of some fixed dimension. Such a numerical vector is known as an embedding of the text document. It is important to note that by document here it can mean a sentence, a question, a paragraph, or an actual document. So we consider the term 'document' in a general sense to be any of the above text collection.
This model has a lot of applications in information retrieval, relevance ranking based on semantic similarity, and document similarity. Once we have these numerical vectors corresponding to the text documents, we can compute the similarity between two documents (vectors) using various measures which would be discussed later. Two major ways to gauge similarity between two text documents is syntactic and semantic similarities.
In syntactic similarity, two text documents are considered similar only if they are structured in a similar way and no importance is given to the context of the

words in the the document. Semantic similarity whereas considers the context of the words occurring in the document and also the entire context (meaning) of the document. Hence, any embedding which captures the semantic context of the document is much more richer than the embedding which only captures the syntactic meaning (structure). A clear advantage of semantic based similarity over syntactic similarity can be seen in example 1.

Example 1: Consider two sentences:
Sentence 1: What is your age?
Sentence 2: How old are you?
These two sentences will be considered very similar based on the semantic similarity but not similar at all based on the syntactic similarity.

6.2 Similarity measures

There are various measures of similarity. Given two numerical vectors, any of the below similarity measures can be used:

- Cosine similarity
  This similarity is the cosine of the angle between the two vectors. It is computed by taking the dot product of normalized numerical vectors. An important consideration is that the angle between the two vectors is always between $0°$ and $180°$. By the property of cosine, we know that $\cos(0°) = 1$ and it goes on decreasing till $180°$ at which $\cos(180°) = -1$. So if two vectors are very close, naturally the angle between them will be close to 0 and hence the cosine similarity will be close to 1. The cosine similarity is given as:
  $$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$
- Euclidean distance
  This is the most well known distance metric, which is computed by the

following formula:

$$d = \sqrt{\sum_{i=1}^{D} (A_i - B_i)^2}$$

where A and B are the two vectors, each of dimension D. The more far apart the two vectors are, more will be the euclidean distance and the more closer the two vectors are, less will be the euclidean distance between them.

- Manhattan distance
  This is another distance metric which is computed by the formula:

$$d = \sum_{i=1}^{D} | A_i - B_i |$$

where A and B are the two vectors, each of dimension D. The relation between the distance and the similarity is the exact same as that for euclidean distance.

We have talked about how to find the similarity between two numerical vectors. Now we will talk about how we convert the text documents into these numerical vectors. As mentioned earlier we need to capture the semantic meaning of the sentence into the embedding of the sentence. There are many encoder models for this purpose and we have discussed here two such models.

6.3 USE Model

6.3.1 Introduction

The Universal Sentence Encoder model was introduced by Google, and is one of the best models for semantic analysis, sentiment classification, and various other NLP tasks. The model takes an english text document as input and produces as output an embedding representation of fixed length. These embeddings can then be used to compute semantic similarity scores.

6.3.2 Encoder Architecture

This section provides a high level understanding of how the Universal Sentence Encoder works. The USE model has two encoders, one based on the 'Transformer' architecture and the other makes use of a Deep Averaging Network (DAN). We have used the model which uses DAN because it has a better time complexity than the one using 'Transformer' architecture, and this is the model we will focus on.

6.3.3 High level working

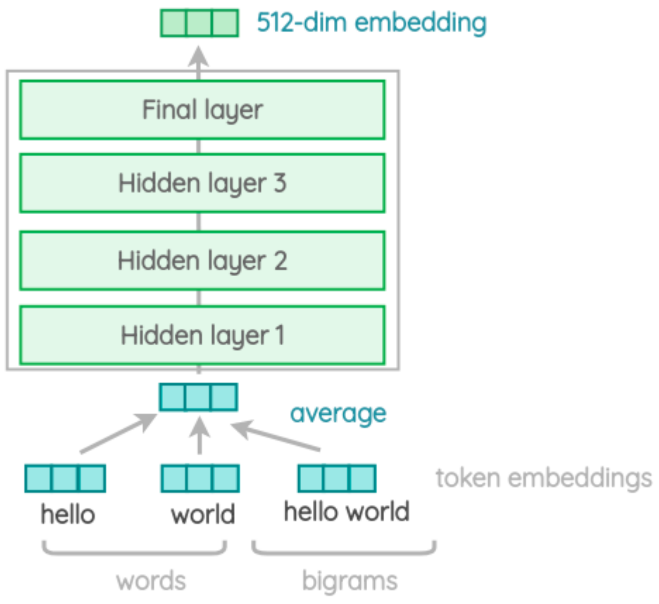
Word embeddings are very popular for NLP tasks, where each word is represented as a feature vector of some fixed dimensions and then words can be distinguished based on these word vectors.

A naive technique to obtain sentence embedding is to average the word embeddings in the sentence and use this average as the final representation of the sentence. But this is not very useful as it does not preserve the complete meaning of the sentence. Let us now see how the USE model works:

1. **Tokenization**: The sentences are first lowercased and tokenized using the Penn Treebank (PTB) tokenizer.
2. **Encode**: The sentences are then encoded into fixed dimension (512) vector. For this as we discussed, we have the encoder model using deep average network. The deep averaging network works as follows:
   (a) The average of embeddings of the words (uni-grams) and bi-grams (known as tokens) in the sentence is computed.
   (b) This average passes through multiple deep neural network feed-forward layers to get the 512-dimension sentence embedding.
3. To learn the sentence embeddings, the encoder is trained on a wide range of supervised and un-supervised data like the SNLI corpus. This SNLI

corpus has pairs of english sentences and one of three possible labels, ‘Contradiction’, ‘Neutral’, and ‘Entailment’, for each pair. If some pair of sentences is somewhat similar, the label is ‘Entailment’, if they are not very related then the label is ‘Contradiction’ and if the relation cannot be established then the label is ‘Neutral’.

The architecture of deep averaging network can be seen below.



6.4 BERT Model

BERT stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers so **BERT** is based on the Transformer architecture. **BERT** has been pre-trained on a large corpus of unlabelled text including the entire Wikipedia(that’s 2,500 million words!) and Book Corpus (800 million words).IT is a “deeply bidirectional” model. Bidirectional means that BERT learns information from both the left and the right side of a token’s context during the training phase.

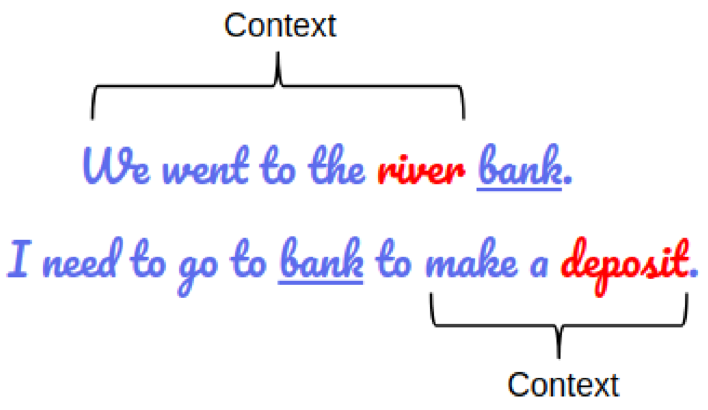
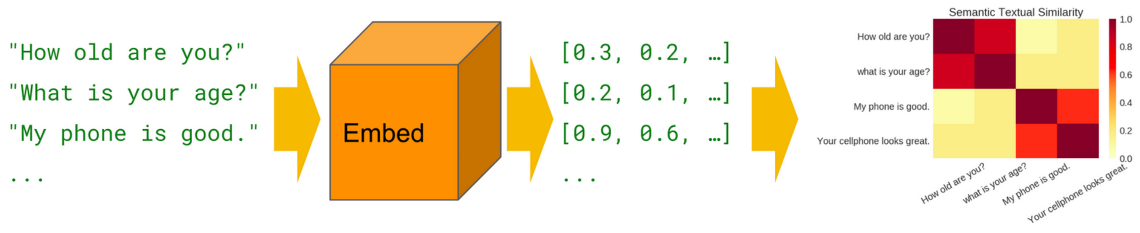

Context
We went to the river bank.
I need to go to bank to make a deposit.
Context

Figure 2: BERT captures both left and right context

If we try to predict the nature of the word "bank" by only taking either the left or the right context, then we will be making an error in at least one of the two given examples.One way to deal with this is to consider both the left and the right context before making a prediction. That's exactly what BERT does.

Every input embedding is a combination of 3 embeddings:

- Position Embeddings
- Segment Embeddings
- Token Embeddings



Sentence similarity is typically calculated by first embedding the sentences and then taking the cosine similarity between them. (Source)

BERT is pre-trained on two NLP tasks:

- Masked Language Modeling.
- Next Sentence Prediction.

1.Masked Language Modeling: In this modeling we mask some words of a sentence and the model is trained to predict what that word is. This is the crux of Masked Language Modeling. Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

1.Adding a classification layer on top of the encoder output.

2.Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.

3.Calculating the probability of each word in the vocabulary with softmax.

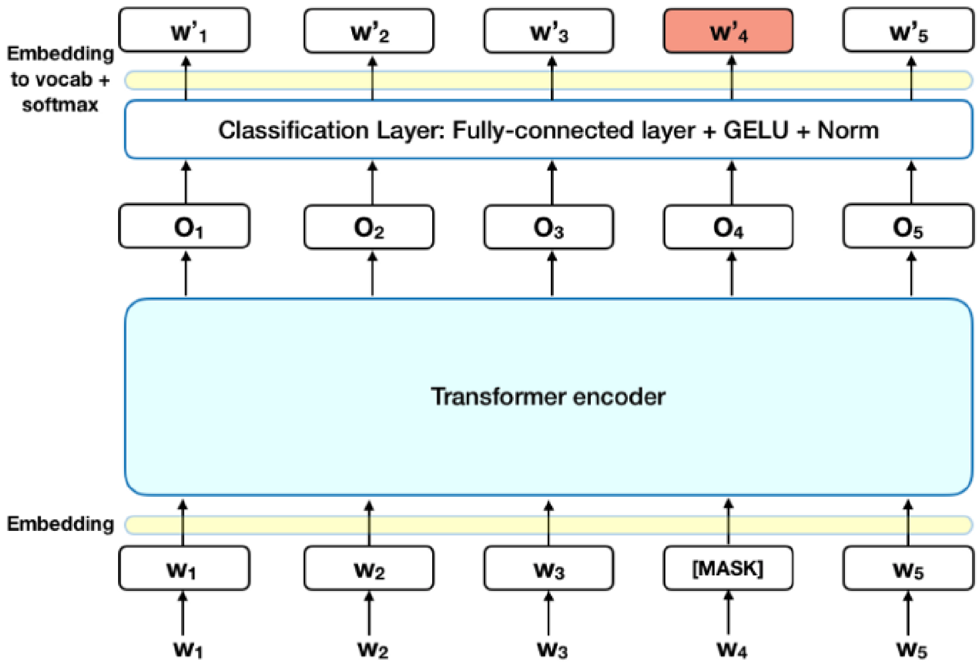
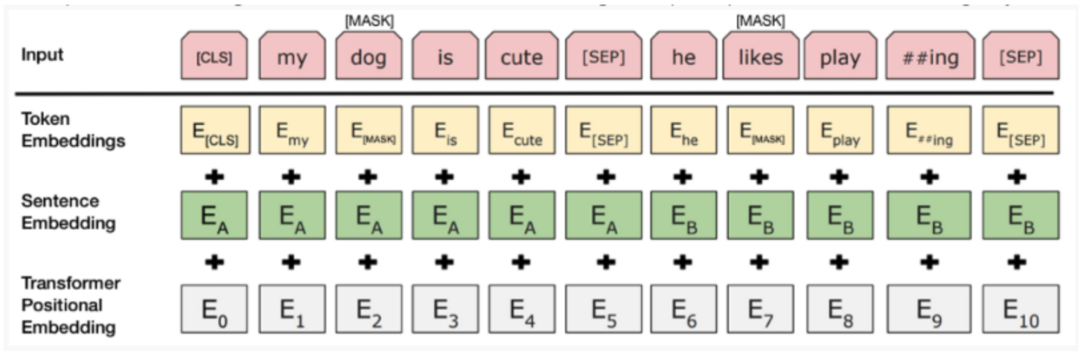

w'₁
w'₂
w'₃
w'₄
w'₅
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O₁
O₂
O₃
O₄
O₅
Transformer encoder
Embedding
w₁
w₂
w₃
[MASK]
w₅
w₁
w₂
w₃
w₄
w₅

Figure 3: BERT captures both left and right context

2.Next Sentence Prediction. In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between two sentences in training, the input is processed in the following way before entering the model:


[MASK]
[MASK]
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

To predict if the second sentence is actually connected to the first sentence, following are the steps that are taken:

1. The entire input chain goes through the Transformer model.
2. The token output [CLS] is converted to a 2×1 vector, using a simple classification class (the matrix has learned about weights and deviations).
3. Calculate the probability of the next sentence with softmax.

This is how **BERT** is able to become a true task-agnostic model. It combines both the Masked Language Model (MLM) and the Next Sentence Prediction (NSP) pre-training tasks.

6.5 SBERT

Two main reason why vanilla BERT is not so useful in sentence semantic comparisons are-

1.Finding the two most similar sentences in a dataset of n This would require us to feed each unique pair through BERT to finds its similarity score and then compare it to all other scores. For n sentences would that result in $\frac{n(n-1)}{2}$.

2.Performing semantic search This task entails finding the most similar sentence in a dataset given a query. Ideally, this would be done by comparing the query to all existing sentences, with for a dataset of n sentences would require n comparisons.

To sum up, asking BERT to compare sentences is possible but too slow for real-time applications. The main culprit is that BERT needs to process both sentences at one in order to measure similarity.

SBERT is a so-called twin network which allows it to process two sentences in the same way, simultaneously. These two twins are identical down to every parameter (their weight is tied), which allows us to think about this architecture as a single model used multiple times.

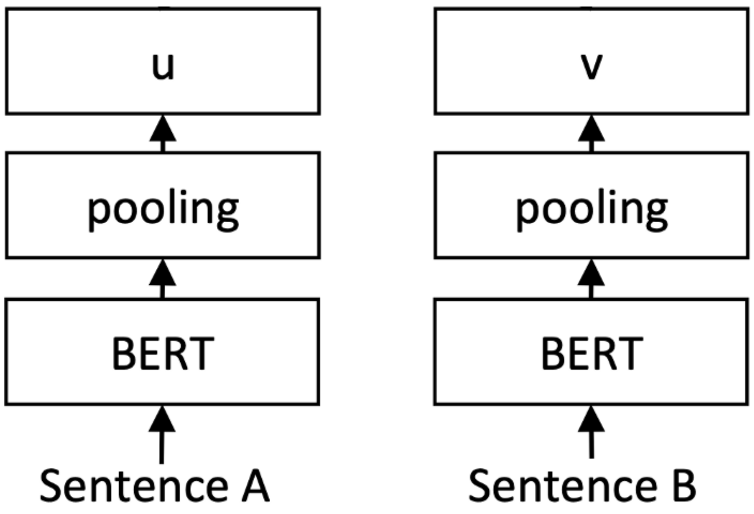


Figure 4: BERT captures both left and right context

It becomes apparent from the image that BERT makes up the base of this model, to which a pooling layer has been appended. This pooling layer enables us to create a fixed-size representation for input sentences of varying lengths

First, the original pair of sentences pass through BERT to embed fixed sized sentences. Then in the pooling layer, mean aggregation, which was proved to have best performance compared to max or CLS aggregation, is used to generate **u** and **v**.

Classification Objective Function We concatenate the sentence embeddings u and v with the element-wise difference $|u{-}v|$ and multiply it with the trainable weight $W^t \epsilon R^{3n \times k}$: where n is the dimension of the sentence embeddings and k the number of labels. We optimize cross-entropy loss.

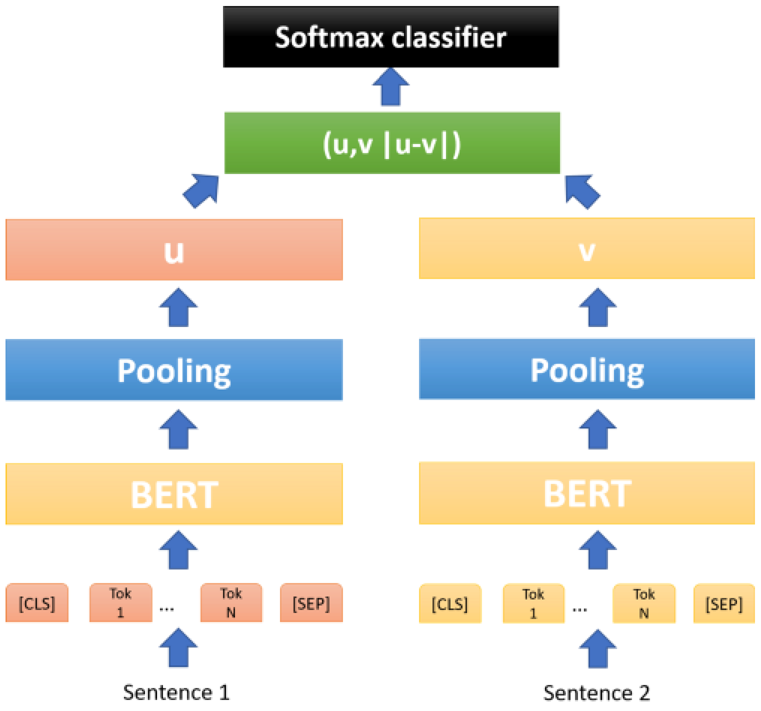


Figure 5: BERT captures both left and right context

softmax(W^t (u, v, $|u - v|$))

Regression Objective Function In case of regression task, we compute sentence embeddings and the cosine similarity of the respective pairs of sentences. We use mean squared-error loss as the objective function. MSE(Wt(u,v,cosine - sim(u,v))

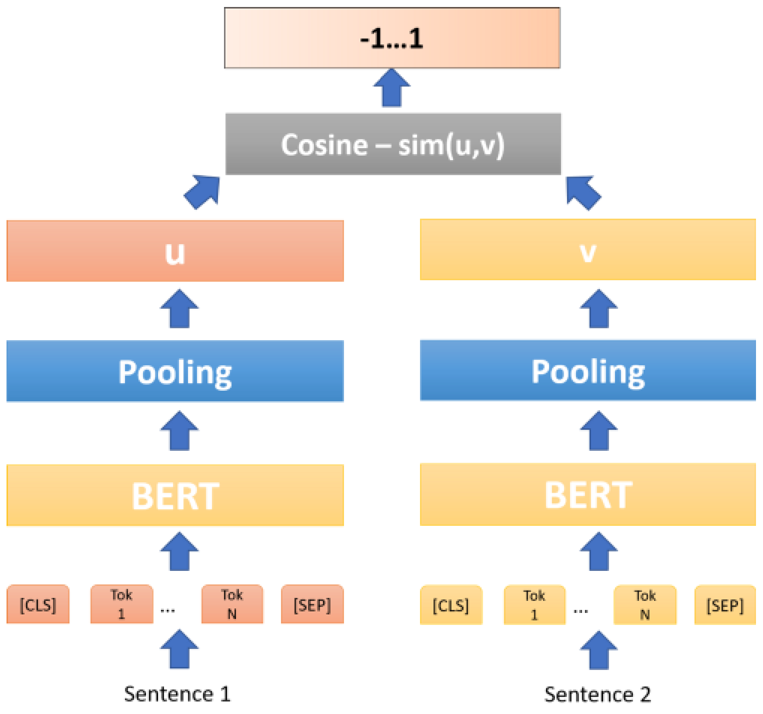


Figure 6: BERT captures both left and right context

7 METHODOLOGY

Initially we tried a variety of models and similarity measures which are now explained in brief along with their challenges and why they did not feature in our final models.

Model 1

One of the first models we tried was a modified Tf-Idf model. In this model instead of using the standard term frequency, we modified it by applying a sub-linear function so that the words occuring multiple times in the same question are weighed-down. We used the standard Idf formula to compute inverse document frequency of the words in the question to weigh down the importance of words occurring in large number of questions. Finally, we also used pivot length normalization to penalize longer questions than the average question length and reward shorter questions as each word in a short question holds more value. The final model equation can be seen below.

$$\frac{\ln\left(1+\ln(1+\mathrm{tf}(\mathrm{w_i},\mathrm{q_j}))\right)}{1-\mathrm{b}+\mathrm{b}*\left(\frac{|\mathrm{q_j}|}{\mathrm{avql}}\right)} * \log\left(\frac{\mathrm{N}+1}{\mathrm{q_{w_i}}}\right)$$

where $\mathrm{w_i}, \mathrm{q_j}$ means $\mathrm{w_i}$ is the $\mathrm{i^{th}}$ word in $\mathrm{j^{th}}$ question, $|\mathrm{q_j}|$ is the length of $\mathrm{j^{th}}$ question, avql is the average question length in the entire corpus, N is the total number of questions in the corpus and $\mathrm{q_{w_i}}$ is the count of questions in which the $\mathrm{i^{th}}$ word occurs. Once we had the vectors for the query and the questions, we used simple dot product for the similarity measure as we had already used the normalization earlier. Once the query and the questions were vectorized, instead of using cosine similarity, we used a simple dot product for similarity as we had already applied a normalization technique.
There were multiple challenges with this technique. First challenge was that even though this model considered the relative importance of each word in the entire corpus, it did not capture the semantic meaning/ context of the question. The second challenge was that since we implemented the model from scratch, we could not optimize it and training the model on a large dataset was computationally infeasible.

Model 2

We consider the normal binary BoW model and compute the similarity based on the cosine similarity. One idea that we thought about was of penalizing the similarity score more if the two questions had very different words. If we consider this case then the standard cosine similarity will be close to '0' as very few words are common, but we further penalize this score (make the model stricter) by deducting the count of number of differing words in the two questions. So the formula for the new cosine similarity score becomes:

$$\mathrm{simscore(A,B)} = \mathrm{simscore(A,B)} - (\mathrm{differing\,words\,in A\,and\,B})$$

where simscore represents the similarity score found by the cosine similarity between two questions and '' denotes 'number of'.

Another idea that we thought of was to learn a suitable similarity score with the help of a training dataset which has similar question pairs. We will compute the similarity scores for each of these pairs and take the average similarity score. Now while testing, if for any query the similarity score with a question comes close to the average score, we will consider them similar. Both of these ideas performed very poorly and hence we do not consider them henceforth.

Similarity Measures We tried three similarity measures, cosine similarity, euclidean distance based similarity and manhattan distance based similarity. Finally we were able to only implement the cosine similarity measure as the other two distance based similarity measures were computationally infeasible to implement. Both the models struggled to work with the two distance based similarity measures and hence we could not go ahead with them.
Finally we have used two models as described earlier in Universal Sentence Encoder and Sentence-BERT. We have also used one similarity measure in cosine similarity. We also tried to combine the two models and computed a weighted average similarity score based on which the output of test questions was predicted. We have presented the results of these three models and the corresponding accuracies. Thus we have compared the three models- Universal Sentence Encoder, Sentence-BERT and a combination of the two models. The overall workflow is as follows:

1. **Embedding the questions in the corpus**

   The embeddings of all the questions in the corpus is computed corresponding to both the models. The Universal Sentence Encoder and the Sentence-BERT output a 512-dimensional and a 768-dimensional embedding for each question respectively.

2. **Computing similarities**

   Using the embedding generated in the previous step, for each test query we compute the cosine similarity with all the questions in our corpus. Thus for each test query we get a 34,235 length similarity vector as we have

34,235 questions in the corpus.

3. **Computing Accuracy**

   Using the similarity vectors generated in the previous step, we now calculate the accuracy for the two models using the test dataset. From the test set we know that a test question has a list of question IDs from the corpus which are actually similar to the test question.
   The method of computing the accuracy for one model- Universal Sentence Encoding is described. The accuracy for the SBERT model will be calculated in the exact same fashion.
   Given the similarity vector for each test question, we retrieve the top 7 most similar questions based on the maximum similarity scores. Once we have these as the 'predicted similar questions list', then from the test data-set we get the 'actual similar questions list'. Consider the i^{th} test question. For this question if there is some intersection between the two lists, it essentially means that the model is predicting at least one duplicate question of this test question correctly. Hence we increment the 'correct' count by 1. We do this for all the test questions and finally divide the correct count by the total number of test questions to get the accuracy.

4. **Answering User Query**

   We found that out of the three models, Universal Sentence Encoder with cosine similarity produced the best accuracy. Hence to answer the user query we have used this model.
   We first take in the user query through a GUI, which is then processed to remove the stop words, punctuation, and other symbols. The user query is then embedded using the Universal Sentence Encoder model. The cosine similarity of this embedded query is then computed with all the question embeddings which we already have from step 1. Then the top 7 similar questions are produced as output to the user along with the links of those questions.

8 RESULTS

This section provides the overall results of the project, which include the comparison of the three aforementioned models, their performance and the corresponding accuracies.

We also present the user question answering GUI. We take in a query question from the user and we display the 7 most similar questions predicted by our model. We also provide a link with each question for the user to visit the question if he/she wishes.

MODEL	**N**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
USE Cosine	31.293%	41.723%	45.805%	50.567%	53.515%	57.823%	60.317%	62.585%	64.172%	65.76%
SBERT Cosine	16.553%	22.449%	25.624%	26.984%	28.118%	29.478%	29.705%	31.066%	32.2%	33.107%
Final model accuracy	31.973%	40.816%	46.259%	49.206%	52.608%	56.009%	58.73%	60.317%	63.039%	63.946%

Table 2: **Accuracy of the three models with varying N**

.

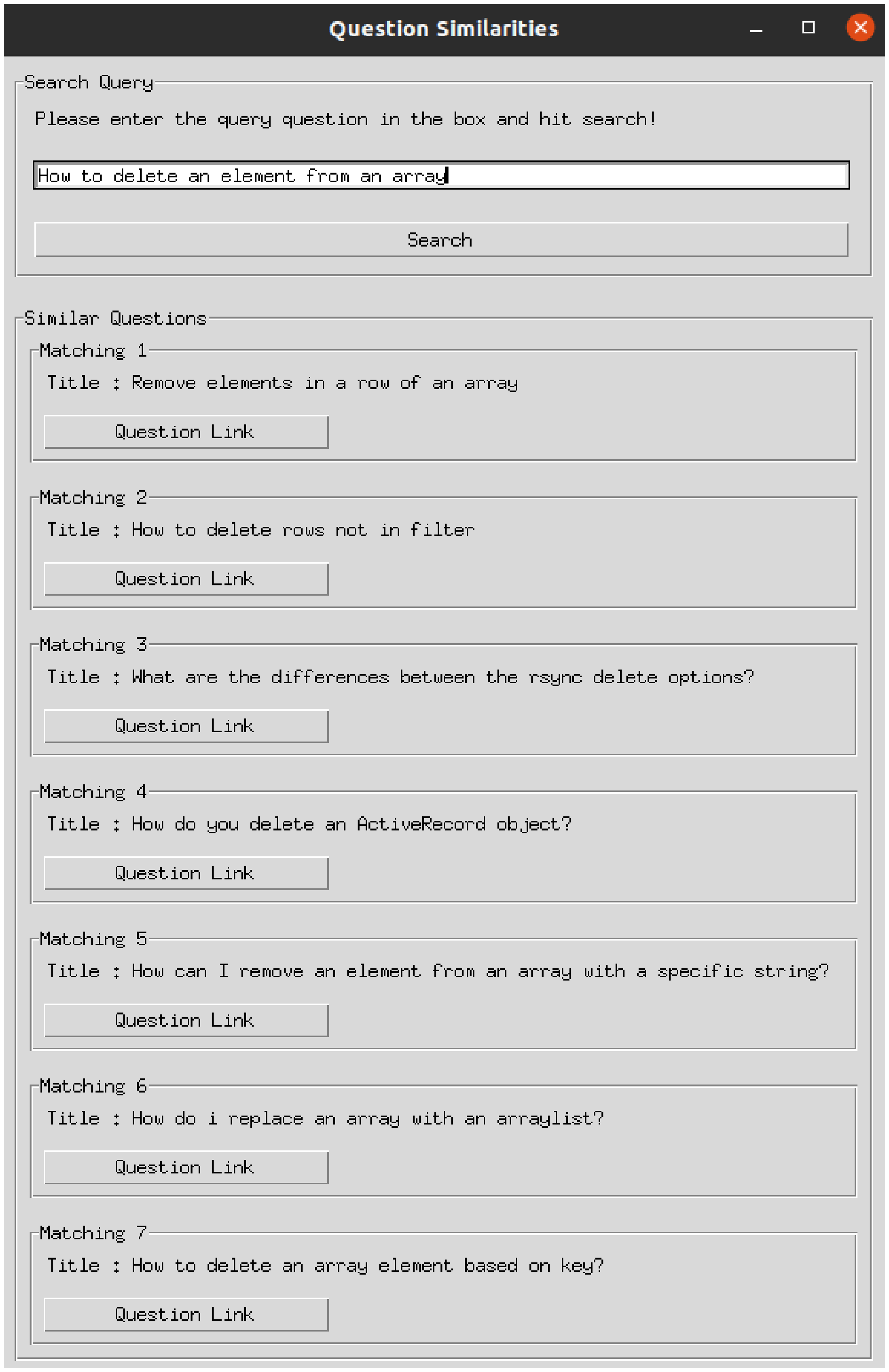
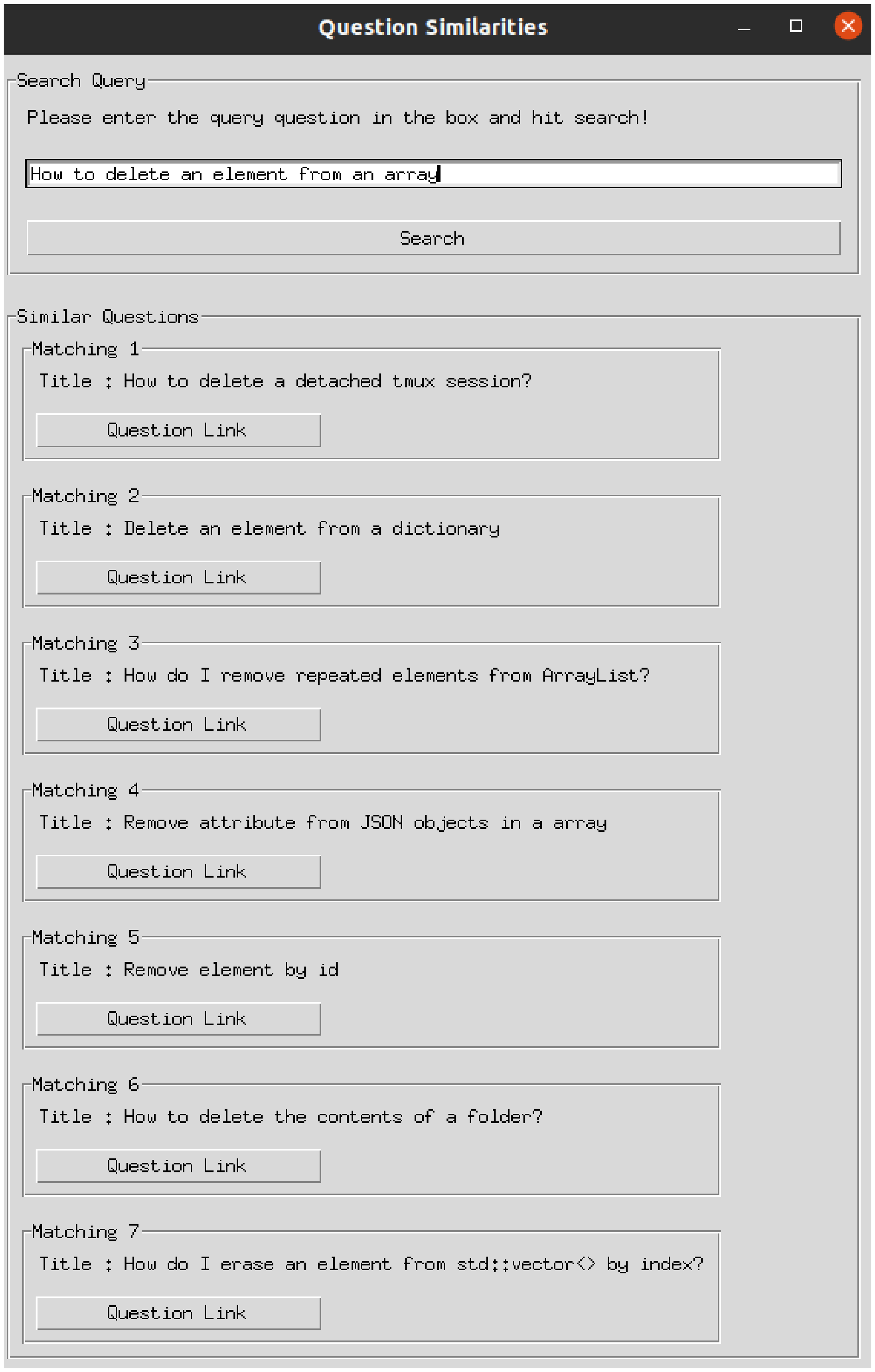

Question Similarities
Search Query
Please enter the query question in the box and hit search!
How to delete an element from an array
Search
Similar Questions
Matching 1
Title : Remove elements in a row of an array
Question Link
Matching 2
Title : How to delete rows not in filter
Question Link
Matching 3
Title : What are the differences between the rsync delete options?
Question Link
Matching 4
Title : How do you delete an ActiveRecord object?
Question Link
Matching 5
Title : How can I remove an element from an array with a specific string?
Question Link
Matching 6
Title : How do i replace an array with an arraylist?
Question Link
Matching 7
Title : How to delete an array element based on key?
Question Link

Figure 7: Most similar 7 questions for the SBERT model with cosine similarity measure.



Please enter the query question in the box and hit search!

How to delete an element from an array

Matching 1
Title : How to delete a detached tmux session?

Matching 2
Title : Delete an element from a dictionary

Matching 3
Title : How do I remove repeated elements from ArrayList?

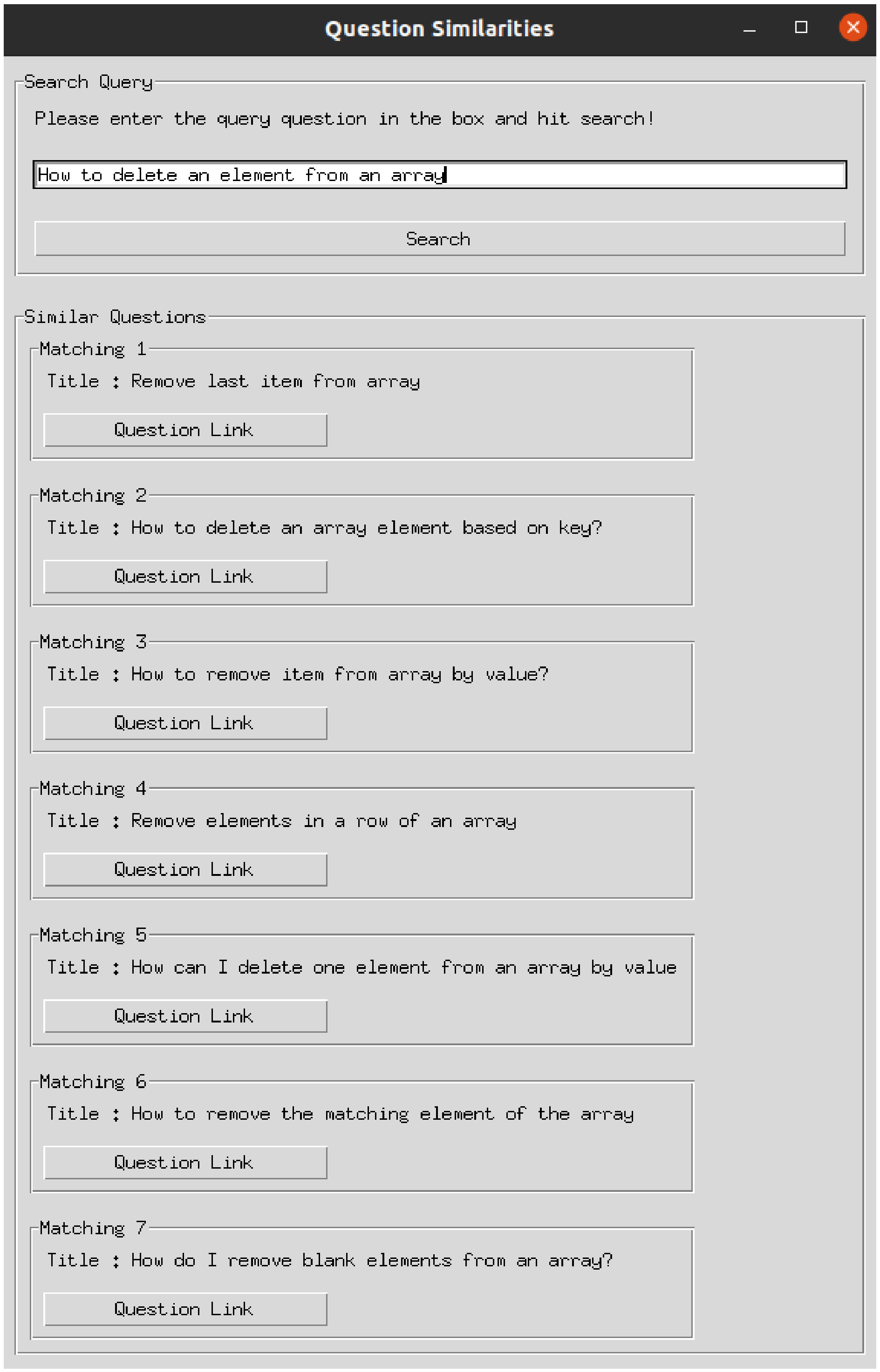
Matching 4
Title : Remove attribute from JSON objects in a array

Matching 5
Title : Remove element by id

Matching 6
Title : How to delete the contents of a folder?

Matching 7
Title : How do I erase an element from std::vector<> by index?

Figure 8: Most similar 7 questions for the SBERT model with manhattan similarity measure.


Question Similarities

Search Query

Please enter the query question in the box and hit search!

How to delete an element from an array

Similar Questions

Matching 1
Title : Remove last item from array

Matching 2
Title : How to delete an array element based on key?

Matching 3
Title : How to remove item from array by value?

Matching 4
Title : Remove elements in a row of an array

Matching 5
Title : How can I delete one element from an array by value

Matching 6
Title : How to remove the matching element of the array

Matching 7
Title : How do I remove blank elements from an array?

Figure 9: Most similar 7 questions for the USE model with cosine similarity measure.

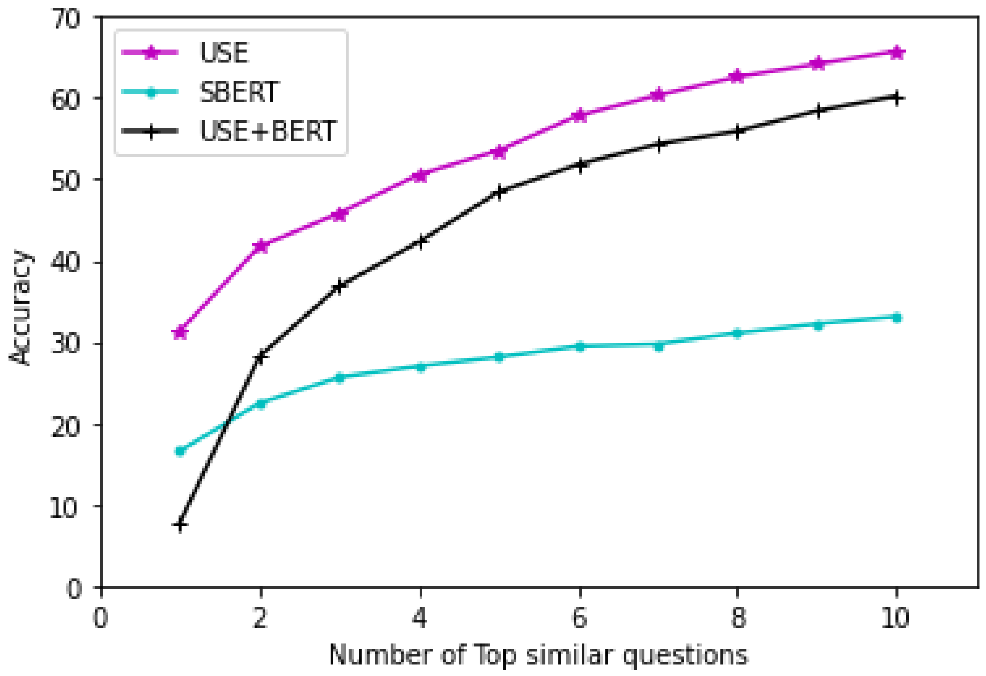


Figure 10: A comparision of the accuracy of the three models using the cosine similarity measure.

9 DISCUSSION

1. Comparision of the three models considered, USE, SBERT and the combined model, is shown in the table (2). We can clearly see that USE has performed better than the other two, and this was precisely the reason why more weight was given to USE model (6/7) as compared to that of SBERT (1/7) while combining the two models. Even after this, the combined model's performance was sub-par than that of the USE model. We can safely conclude that this was due to the fact that SBERT performed very poorly on our dataset and for some test queries it distorted the similarity score hence affecting the combined model's accuracy.

2. We have considered the value of N as 7, as we wanted to have a balance

between the accuracy and the convenience of the user. If we had taken N to be large then although the accuracy would have improved, the user would have been confused by the large number of choices. On the other hand, if we had taken N to be very small, then the user might have been unsatisfied with results of the model.

3. We were able to predict and output top 7 similar questions for any given user query with an accuracy of 60%.

4. Finally we learnt that there is a lot of data available which can be made good use of and a lot of modern problems can be solved with it. After trying many models we can also conclude that when it comes to sentence level similarity checking, it is important to preserve the semantic meaning (context) of the sentence and even extensions and modifications to the standard techniques like the Tf-Idf do not help a lot. We also learnt that any model is data dependent, i.e. it is not necessary that if a model is held in high regards, it will always perform the best for all datasets. Hence it is important to try and experiment with multiple techniques to understand what works best for the given dataset and the given task.

10 FUTURE DIRECTION

This project can be further extended to come up with a **Search Engine** dedicated for the combined dataset of Q&A sites like Stack Overflow, Gate Overflow, Super User, Stack Exchange. At present each of these sites do give suggestions of similar questions for a particular question , but we can extend work of this project to provide a search engine which could give all similar questions that have been asked and answered across all fore mentioned Q&A sites together with various indicators of a well answered question like number of views , number of UpVotes, total answers , most UpVoted answers.
This Search Engine can provide a common and dedicated platform for all users of these Q&A sites to get their queries answered quickly as a similar query might have been already well answered in any one of these platforms.

We have worked on plain text questions in English language in this project which can be made more robust by providing the ability to work with multilingual questions as well as with questions having images , programming codes and mathematical expressions.
Another addition that we can do is, increase the corpus and take questions from more sites. This will enable the model to predict similar questions for a wide range of user queries. Currently, the model can do well only for questions which are technical as our dataset is collected from sites which offers such questions only.

11 IMPORTANT LINKS AND REFERENCES

GitHub link for the project
(https://github.com/anup00900/datamining_project)

[1] StackOverflow: (https://stackoverflow.com)

[2] SuperUser : (https://StackUser.com)

[3] StackExchange: (https://StackExchange.com)

[4] Gateoverflow : (https://gateoverflow.in)

[5] MathOverflow :(https://mathoverflow.net)

[6] Overleaf :(https://overleaf.com)

[7] TowardsDataScience:(https://towardsdatascience.com)

[8] Flowchart online tool:(https://app.diagrams.net)

[9] Singhal, Amit, Chris Buckley, and Mandar Mitra, 1996a, Pivoted document

length normalization. In Proc. SIGIR, pp. 21-29. ACM Press.

[10] Is Question Answering an acquired skill?, Ganesh Ramakrishnan, Soumen Chakrabarti, Deepa Paranjpe, and Pushpak Bhattacharyya. WWW2004, New York City

[11] Regina Barzilay and Kathleen McKeown. Extracting paraphrases from a parallel corpus. In Proceedings of the ACL/EACL, Toulouse, 2001

[12] Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (2002)."BLEU: a method for automatic evaluation of machine translation" in ACL-2002: 40th Annual meeting of the Association for Computational Linguistics pp. 311–318

[13] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

[14] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,Nils Reimers and Iryna Gurevych

[15] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil. Universal Sentence Encoder. arXiv:1803.11175, 2018.

[16] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daume III. 2015. Deep unordered composition rivals syntactic methods for text classification. In Proceedings of ACL/IJCNLP.