DupPredictor: StackOverFlow Duplication Question Detector

Team 18

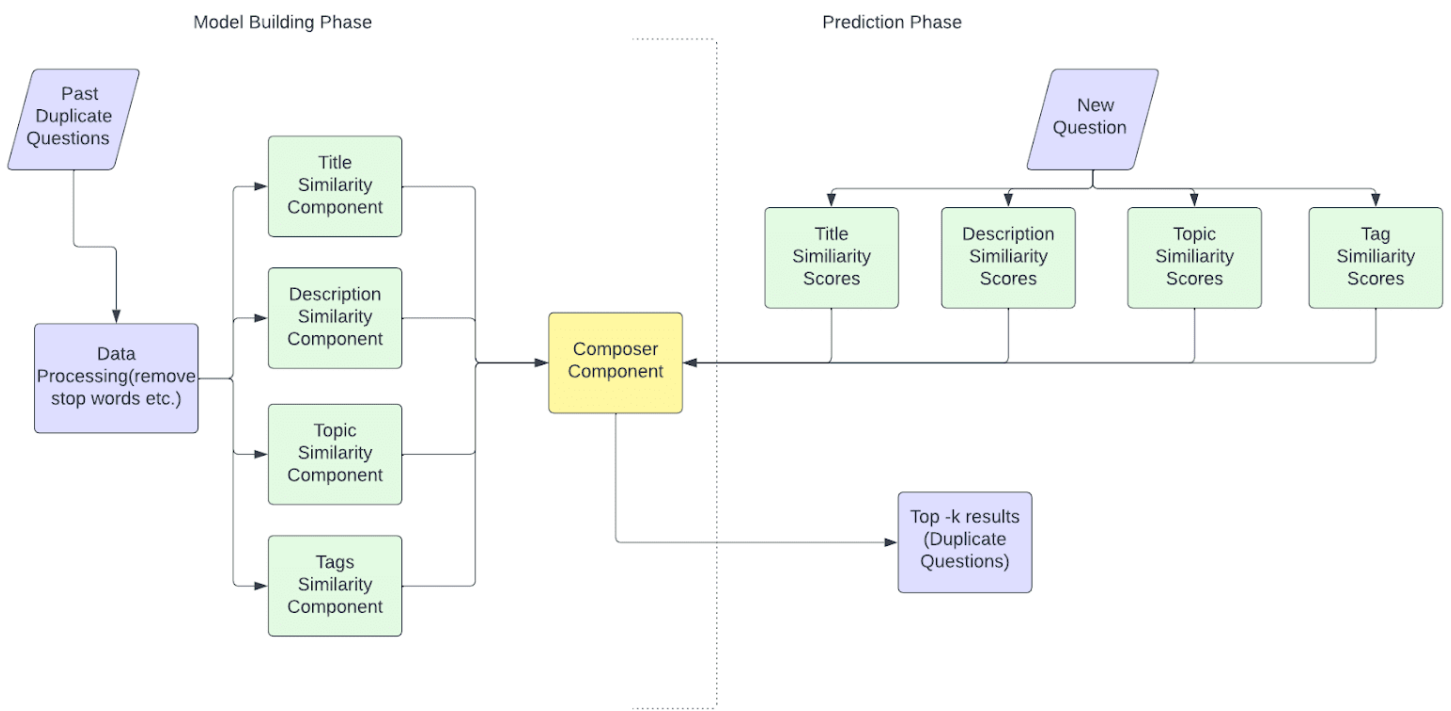
Team Members: Aarush Jain , Abhishek Sharma , Aryan Singhal , Vaibhav Agarwal

Final Report

Stack Overflow is a popular question answering site that is focused on programming problems. Despite efforts to prevent asking questions that have already been answered, the site contains duplicate questions. Stack Overflow recommends that users search previous posts before asking a new question. This is to avoid asking a question that already has been asked and that may already have been answered. Stack Overflow also suggests links to questions whose title matches the new question

Framework

The framework contains two phases: model building phase and prediction phase. In the model building phase , goal is to train a model from past duplicate questions which have been detected.



Dataset

The data is obtained using queries on StackExchange. The query returns 50k rows with original question as PastQues and Duplicate question as DuplicateQues. The query is run on https://data.stackexchange.com/stackoverflow/queries.

```
SELECT
  DISTINCT TOP 50000 p.Id AS PastQuesId,
  p.CreationDate,
  p.Title as PastQuesTitle,
  p.Body as PastQuesBody,
  p.Tags as PastQuesTags,
  pd.Id as DuplicateId,
  pd.Title as DuplicateTitle,
  pd.Body as DuplicateBody,
  pd.Tags as DuplicateTags
FROM PostHistory ph
  JOIN Posts p ON p.Id = ph.PostId
  JOIN Posts pd ON pd.Id = CAST(JSON_VALUE(ph.Text,
'$.OriginalQuestionIds[0]') AS nvarchar)
WHERE
  p.PostTypeId = 1 -- Question
  AND ph.PostHistoryTypeId = 10 -- Close event
  AND ph.Comment = 101 -- Dupe
  ORDER BY pd.Id ASC
```

The dataset looks like:

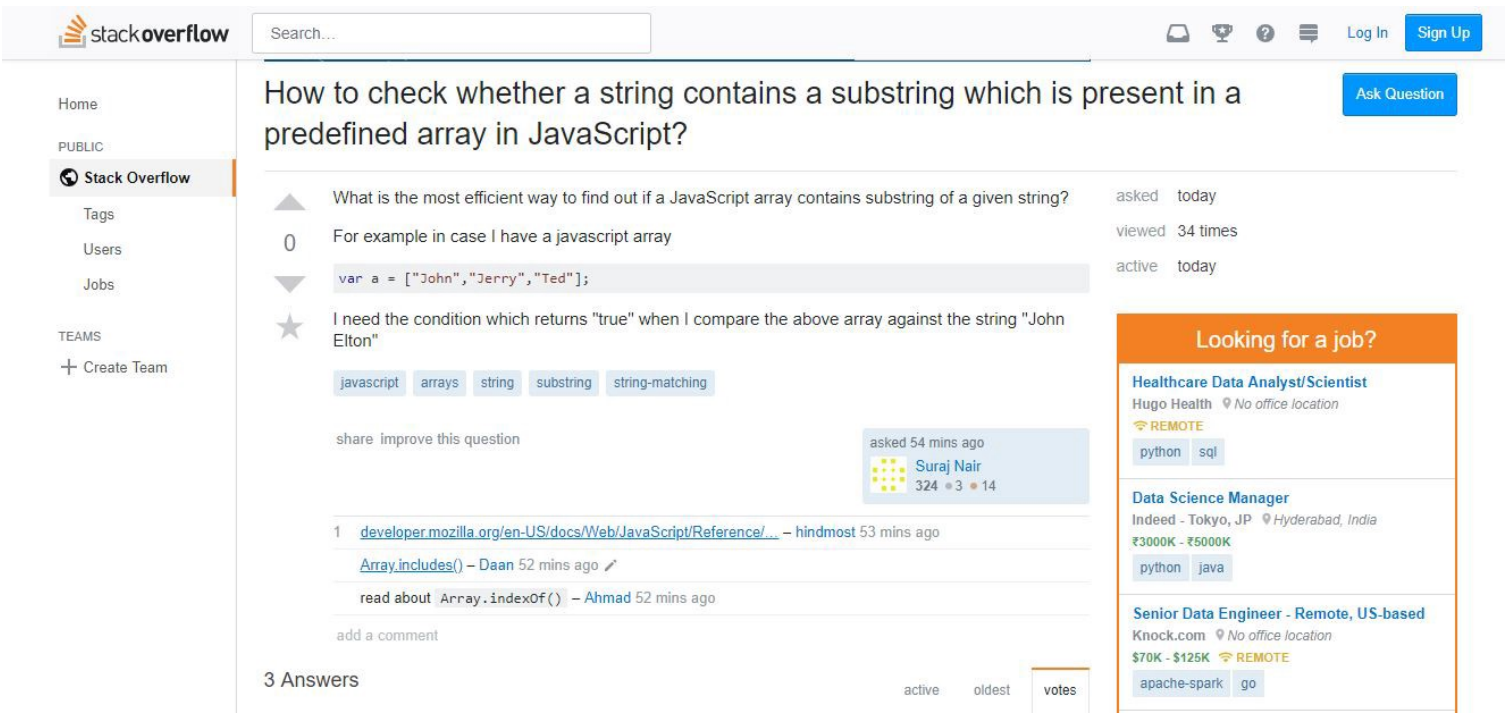
	PastQuesId	PastQuesTitle	PastQuesBody	PastQuesTags	DuplicateQuesId	DuplicateQuesTitle	DuplicateQuesBody	DuplicateQuesTags
0	205853	Why would a JavaScript variable start with a d...	I quite often see JavaScript with variables...	<javascript><naming-conventions>	846585	What is the purpose of the dollar sign in Java...	The code in question is here: \n\n<pre>...	<javascript><naming-conventions>
1	104799	Why aren't Java Collections remove methods gen...	Why isn't <a href="http://java.sun.com/java...	<java><generics><collections>	857420	What are the reasons why Map.get(Object key) i...	What are the reasons behind the decision to...	<java><generics><collections><map>
2	255815	How can I fix my regex to not match too much w...	I have the following line: \n\n<pre><cod...	<regex><perl><parsing><greedy><regex-greedy>	22444	My regex is matching too much. How do I make i...	I have this gigantic ugly string: \n<pre...	<regex>
3	190740	Setting ruby hash .default to a list	I thought I understood what the default met...	<ruby><hashmap>	2698460	Strange, unexpected behavior (disappearing/cha...	Consider this code: \n\n<pre><code>h = H...	<ruby><hash>
4	256277	What is a good reference documenting patterns ...	"C Interfaces and Implementations" shows s...	<c><design-patterns><include>	1804486	Should I use #include in headers?	Is it necessary to <code>#include</code> so...	<c><c-preprocessor><file-organization>
...	...	...	...	...	...	...	...	...
						Why does this CSS	I try to add	

In this example,

Title: "How to check whether a string contains a substring which is present ina predefined array in javascript?"
Body : What is the most efficient way to find out if a JavaScript array contains substring of a given string? For example in case I have a javascript array

var a = ["John","Jerry","Ted"];

Tags: javascript, arrays , strings , substring , string-matching



Data Preprocessing

Body of the questions consists of html tags , punctuation marks and stopwords etc.

STEMMING: Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

In the below code , first all the stop words are removed from the text and than punctuation marks and contract strings are removed and final string is returned.

Tokenisation and Stemming will be done for Title and Body.

```
def TokStemming(text, j):  # Tokenise and Stem the given text
    words = [word for word in text.split() if word.lower() not in
sw_nltk]
    # bag of tokenised words which are not stopwords
    final = []
    for w in words:
        if w not in punctuations and w not in contract_strings:
            final.append(ps.stem(w))
    if j == 1:  # give a string
        return " ".join(final)
    return final
```

REMOVE TAGS: Tags in the dataset are inside brackets which has to removed to convert direct strings. So all the words in Tag list are converted without tags. For example, = javascript.

```
def rmTags(text):
    text2 = text.replace('<', '')
    text3 = text2.replace('>', ' ')
    return text3
```

Similarity Scores

This component computes the similarity between the titles of a pair of questions based on common words that they share. After preprocessing, the titles of two questions are transformed into two bags (i.e., multisets) of words. For two questions m and n, we represent the two bags of words that are extracted from their titles as TitleBag_m and TitleBag_n respectively. Next, we merge T itleBagm and T itleBagn and eliminate duplicate words to obtain the union set T itleBagu, which contains v words. Following vector space modeling , we represent the two titles as two vectors: TitleVec_m= (wt_m1, wt_m2, . . . , wt_mv) and TitleVec_n = (wtn,1, wtn,2, . . . , wtn,v). The weight wtq,i in these two vectors denotes the relative term frequency of the i-th word in question q's title, which is computed as follows:

$$wr_{q,i} = n_{q,i} / \sum_{v} n_{q,v}$$

We measure the similarity between two questions' titles by computing the cosine similarity[10] of their vector representations TitleVec_m and TitleVec_n as follows:

$$SimilarityScore(Title) = TitleVec_m . TitleVec_n / |TitleVec_m||TitleVec_m|$$

```
def cosine_similarity(a, b):
    c = list(set(a).union(set(b)))
    freq_a = np.array([a.count(i)/(a.count(i)+b.count(i)) for i in
c])
    freq_b = np.array([b.count(i)/(a.count(i)+b.count(i)) for i in
c])

    final = np.dot(freq_a,
freq_b)/(np.linalg.norm(freq_a)*np.linalg.norm(freq_b))
    return final
```

Similarly the cosine simliarity scores are calculated for **tags** and **body** also.

For **Topic Modelling** , LDA(Latent Dirichlet Allocation) which classifies or categorises the text into a document and the words per topic. It assumes that documents with similar topics will use a similar group of words. This enables the documents to map the probability distribution over latent topics and topics are probability distribution.

In the algorithm, Body and Title of the question are concatenated and given as input parameter in two formats. One is document to bag of word collection and one is indexed mapping of those words with the topic.

```
#train lda model
body_title = []
for i in range(N1):
    body_title.append((past_df.loc[i, 'PastQuesBody']+past_df.loc[i,
'PastQuesTitle']).split())

id2word = corpora.Dictionary(body_title)

corpus = []
for text in body_title:
    corpus.append(id2word.doc2bow(text))

lda_model = gensim.models.LdaMulticore(corpus=corpus,
id2word=id2word, num_topics=100, minimum_probability=0.0)

lda_score_past = []
lda_score_dup = []
for i in range(N1):
    corpus_past = id2word.doc2bow((past_df.loc[i,
'PastQuesBody']+past_df.loc[i, 'PastQuesTitle']).split())
    lda_score_past.append(np.array(lda_model[corpus_past])[:, 1])
for i in range(N2):
    corpus_dup = id2word.doc2bow((dup_df.loc[i,
'DuplicateQuesBody']+dup_df.loc[i, 'DuplicateQuesTitle']).split())
    lda_score_dup.append(np.array(lda_model[corpus_dup])[:, 1])
```

Training

To Train the dupPredictor model, we assumed our original questions as the questions dataset, and out of all the duplicate questions, we used the first 300 questions as train data for training the composer model parameters, (Title, Tag, Body, and Topic similarity scores).

During training , the model will calculate four parameters a,b,c,d . There are two approaches used for finding the parameters.

Steps before training approaches:

- First store all the LDA probability distributions of words in two separate arrays (Duplicate question and Past question).
- Store all the cosine similarity scores in a matrix for all the combination of duplicate questions and past questions . Matrix will be size of (n1,n2,4) where n1 is size of past question training set ,n2 is the size of duplicate question training set and 4 is the number of all scores (title, tag, body and topics).

First approach presents the pseudocode of our sample-based greedy method. In this method, 10 divisions between 0 and 1 are made and all combinations of a,b,c,d are iterated to find the best parameter values for the model.

```
divisions = 10
maximum = 0
possibility = np.linspace(0, 1, divisions)
best = np.array([0, 0, 0, 0])
for abcd in
itertools.product(possibility,possibility,possibility,possibility):
    if sum(abcd) == 0:
        continue
    cur_val = evaluate(abcd, 1)
    if cur_val>maximum:
        maximum=cur_val
        best=abcd
print(best, maximum)
```

Second approach is using initially random values for a,b,c, and d. This approach involves selecting random numbers for a, b, c, and d in doing 100 iterations. We loop until we reach 1 while keeping the other parameters fixed, changing one parameter's value by 0.01 for each iteration (random). In these iterations, the maximum score parameter is present.

```
iter,alpha,k = 100,0.01,1
final_abcd = np.array([0, 0, 0, 0])
final_score = 0
iter_score = []
for i in range(iter):
  abcd = np.random.rand(4)
  maximum = evaluate(abcd, k)
  for j in range(4):
    best_j = abcd[j]
    abcd[j]=0
    while(abcd[j]<=1):
      cur_val = evaluate(abcd, k)
      if cur_val>maximum:
          maximum=cur_val
          best_j=abcd[j]
      abcd[j]+=alpha
    abcd[j] = best_j
  iter_score.append([abcd, maximum])
  if maximum>final_score:
    final_score = maximum
    final_abcd = abcd
print(final_abcd, final_score)
```

K	a	b	c	d	Total(300)
5	0.77	0.96	0.96	0.28	212

K	a	b	c	d	Total(300)
	0.99	0.55	0.933	0.27	215
	0.98	0.96	0.97	0.3	214
	0.99	0.99	0.6	0.194	224

K	a	b	c	d	Total(300)
10	0.99	0.85	0.59	0.08	241
	0.93	0.695	0.97	0.1	233
	0.94	0.83	0.93	0.2	236
	0.99	0.87	0.39	0.11	245

K	a	b	c	d	Total(300)
20	0.88	0.79	0.99	0.19	257
	0.98	0.97	0.98	0.2	259
	0.99	0.90	0.91	0.18	259
	0.42	0.73	0.25	0.04	267

For three values of K , the evaluate criteria counts the number of correct matchings in the dataset. In the above tables, it can be observed that value of d is very small that shows it has very least role in the mapping with the question. The value of other three parameters are kind of equally important . For recall rates for different values of k , the testing is done on different 500 duplicate questions and 2000 questions as the past questions for making LDA model .

Testing

Testing is done using next 500 duplicate questions for different values of k where k is the number of top scorer questions. Data preprocessing is done for every duplicate question and for every duplicate question , compose score is calculated with past questions data and maximum k scorers questions are observed.

Tables with one of a,b,c removed from the evaluation criteria and recall rates for different values of k.

The best analysis is done using 100 iterations and increasing a,b,c,d by 0.1 and output recall rate results are following :

a	b	c	d	RR@5	RR@10	RR@20
0.76	0.98	0.29	0.07	0.57	0.64	0.73
0	0.95	0.53	0.07	0.38	0.41	0.45
0.56	0	0.49	0.09	0.40	0.49	0.59
0.953	0.99	0	0.06	0.43	0.53	0.62

Observation: Title Similarity Score has very high impact on total score because value of RR@20 is 0.45 which shows total score to be very low . It shows a > b > c > d has contribution on calculating the total score.

Interface

The interface, which has four input boxes—one for each of the following: title, tag, body, and K—is made using the Tkinter Library. The user will enter these values, press "Enter," and then submit. Top-k matching questions will be displayed in the output box.