CMPT 732 (Fall 2022)

Duplicate Question Analysis and Inference on StackOverflow Data

Hassan Ahamed Shaik, Dilip Reddy Basireddy, Nagendra Reddy Vippala, and Rubin Hazarika

---

1. Problem Definition

Question and answer forums like StackExchange, Quora and Reddit are hubs for gathering, storing, and sharing knowledge on a multitude of topics. Given the heavy traffic on these sites, it can be difficult for moderators and users to monitor and filter through the influx of posts. This project targets a specific problem that arises from the large inflow of unmonitored data on these forums: duplicate questions. Our work focuses on StackOverflow, a forum for software developers and programmers, that currently relies on vigilant, high-reputation users to report duplicate questions. While this manual method is effective, it is laborious and slow.

Our project includes extracting duplicate and non-duplicate questions from StackOverflow data. Using that data, we have performed data analysis and we have trained two models that can 1) detect if two questions are duplicates and 2) search for similar questions over a large corpus of data.

2. Methodology

1. Data extraction:

All the data for this project is extracted from: "https://archive.org/download/stackexchange". Of the available data, we only used the Posts, PostLinks, Users and PostTags tables.

The files containing the tables were large and in a non-binary xml format. We used PySpark along with a spark-xml jar to read those tables and convert them to a binary ORC format. This change in formatting, reduced file size and the read time significantly (from hours to mins).

2. Data Analysis and Visualization:

To begin with, we analyzed the available data by splitting it up and cleaning it up using spark. After preparing the data for processing, we used PySpark to group the data by year and tag, this speeds up further analysis.

Using PySpark we have executed multiple queries to get the data required for visualizations (PySpark query code is added to github), we have used plotly and streamlit to build visualizations. Additionally, we have used Tableau to perform tag-based analysis to build interactive visualizations like word clouds and bubble chart.

Primary analysis included finding total duplicate questions, tags and users, we also did some time series analysis to find the rate of duplicate questions per tag over 10 years of time period. Along with this, we analyzed the average time taken for questions to be marked as duplicate.

In addition to these graphs, we have used google maps API and users table to build a location based heat map which depicts the number of questions asked from each country in big data and Machine learning tags.

3. Data Preprocessing:

First part in preprocessing is generating a dataset for training the Ml model, for that we have used data from Posts and PostLinks tables.

PostLinks Scheme: (we have only taken rows which had "3" has LinkTypeId)

Question-1 Id	Question-2 Id	LinkTypeId
Id of source question	Id of target question	“1”-Linked “3”-Duplicate

Posts Table has many attributes out of which we have selected “Id”,”Title” columns and joined them to Postlinks Table, so that we can get question titles which are marked as duplicate.

To get questions which are non-duplicate we have randomly taken more than 200,000 questions from posts whose LinkTypeId is not 3 and 1.
Overall we have extracted 200,000 sets of duplicate and non-duplicate questions as a data set to train the model.

Similarity Classifier model:

Feature engineering and data cleaning:
Converting to lower-case, removing stop words excluding interrogative words, removing punctuations, finally lemmatizing words using the Spacy NLP models.

After performing analysis on multiple custom features we have choose five features to embed to questions:

- Common_word_count_ratio: It is a ratio of common lemmatized words between two questions to total number of distinct words in both questions
- Sorted_fuzzy_ratio: First we have sorted words of both questions in alphabetical order, then using fuzzywuzzy we have generated fuzzy ratio of sorted questions(levitain distance)

SBERT Embedding:
We have used pretrained distil_bert from sentence transformers to embed both questions to form vectors of size 768 dimensions, this model uses mean pooling to form sentence embeddings.
After generating embedding we have used cosine similarity, manhattan distance and euclidean distance to calculate similarity metrics between two questions, from our analysis we found out that for SBERT cosine similarity is yielding the best results.

Universal sentence encoder (USE):
Using only one model embedding did not give good results so, we added universal sentence encoder embedding(model from google), speciality of this model is, it has been pretrained on multiple downstream tasks like question answering, natural language inference, this helps top vectors which represents the model better.For USE we have used both cosine similarity and euclidean similarity because these metrics provided complementary information that covered a wide dataset.

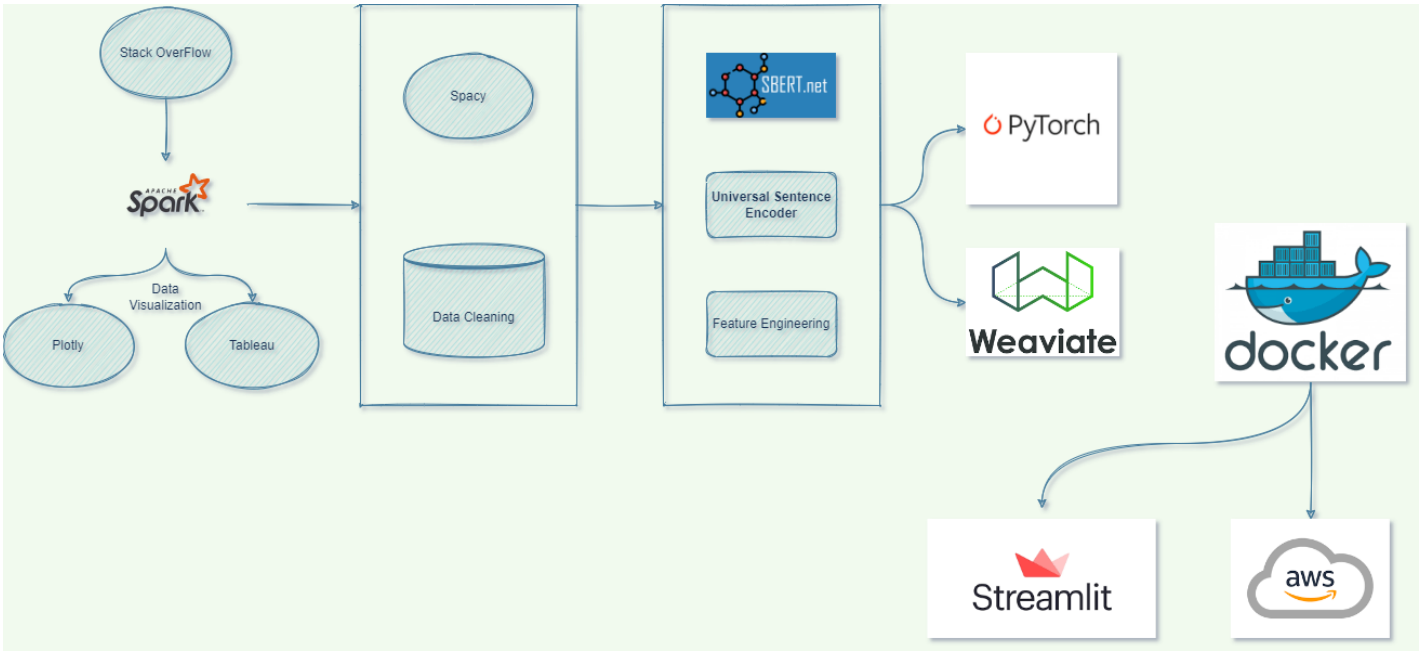
Neural Network:
We have used pytorch to build a MLP model with two layers with relu and sigmoid activation functions. Adam optimizer was used to train the model.
After 50 epochs we got 89 percent training and testing accuracy and model.

5. Semantic search
This pipeline is used to search similar questions in a large corpus of data. We have used questions from python,ML, big data,facebook and image processing tags to build a corpus of 60,000 questions and these questions are converted to vectors using the USE model. Further these questions are uploaded to Weaviate for searching.

Weaviate is a low-latency vector search engine used to store text and its embeddings. In Weaviate, data objects are connected using graph structures and efficient vector indexing algorithms are used to access the data faster. Weaviate typically performs nearest neighbor searches of millions of objects in considerably less than 100ms. Weaviate backed can be accessed through docker container and this pipeline is highly scalable and fault tolerant.

6.Front-end

Streamlit is used to build the front-end for this application. Streamlit app and data required for visualizations are packaged into docker image and this image is uploaded to AWS ECR. ECS cluster and EC2 nodes are used as compute resources for scalability and availability so that users can access this application from anywhere and compute resources can be scaled based on the usage.



3. Problems and Feature scope

Our initial dataset size was more than 80Gb and working with that was very time consuming, so we have used PySpark to read and group data(year and tag wise) as required for visualizations, we have used ORC format to store data, this made the process very efficient.

Initially we faced problems generating location based heatmap because the format of the location data in the Users table was not correctly formatted. For this we have used Google map api provided by GCP to extract countries from a given location.

One of the biggest problems we faced is converting question text into embeddings, for this we have initially tried word2vec but because many rare software words in stackoverflow data word2vec did not work, later we used fasttext but its embeddings were weak. Using language models like SBERT and USE which are specifically pre trained one very large text datasets provided good accuracy.

As part of feature engineering many simple features like absolute length difference did not separate the labels so we have used complex features like sorted fuzzy score.

Scalability and performance is important while working with large datasets so we have been using technologies like Weaviate and AWS EC2.

We have faced many dependency issues, to solve these issues we have carefully organized our code using virtual environments and git version control.
As a feature scope we can train custom cross encoders, this can further improve accuracy(requires powerful compute resources). As the size of data increases we can connect the EC2 instance to AWS S3.

4. Results

4.1 Dataset analysis

We conducted some preliminary analysis on the processed data to analyze the distribution of duplicate and non-duplicate questions in the dataset used for model training. Our model was trained on a relatively balanced dataset to avoid bias. A majority of the questions (> 91%) were unique to the dataset and did not have repeats - a good indicator of a diverse training set.

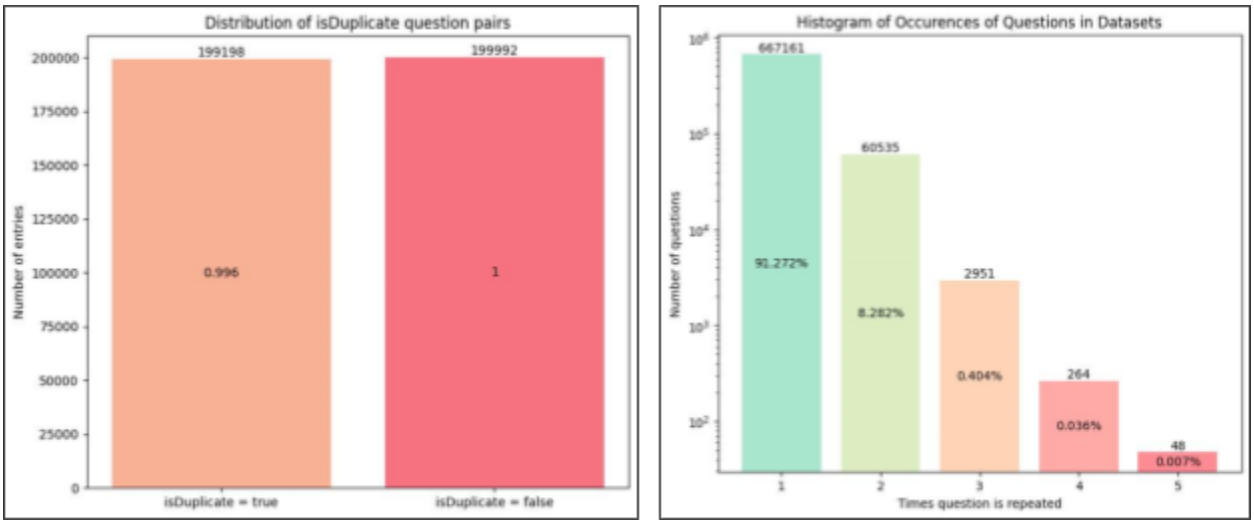


Figure 1 - Distribution of duplicate and non-duplicate questions across model training data.

As part of the front-end, we integrated a series of interactive data visualization tools to demonstrate the scale and distribution of the full dataset. A few key analyses from the plots (found on the Streamlit front-end, in the Dashboard):

- While our model was trained on a balanced subset of the data, the full dataset acquired from StackOverflow is heavily skewed. Only 3% of the posts are duplicate.
- Over the three years observed (2018 - 2020), the Javascript tag was present in the largest number of questions. However, the C++ tag had the largest ratio of duplicate questions to questions asked - 5.3% of questions associated with C++ were duplicate.
- Over the last 3 years, PHP, Java and HTML have consistently been in the top 4 most duplicated tags.
- The percentage of duplicate questions for each of those three tags has been increasing over the last three years, further demonstrating the necessity of an automated system for detecting duplicate questions
- On average, questions take more than 24 hours to be flagged as duplicate, further proving the necessity of automated duplicate detection.
- The outliers in the time series dataset (time to flagged duplicate > 400 hours) have grown more frequent between May 2019 and May 2020. It would be interesting to see if this increase in frequency continues over the upcoming years and keeps pace with the increased volume of duplicate questions.

- The USA, Germany and India are standouts for the volume of questions asked under the tags: ‘big data’ and ‘machine learning’. Each country is the leader in their respective continents for the number of questions asked.
- Accounting for the number of users in each country, (USA = 28312, India = 105626, Germany = 46966)[1], we can see that the USA leads in the number of questions per user, followed by India and then Germany.

The data presented as part of the ‘Tableau Visualization’ section of the Streamlit UI, offers us a good look at the distribution of tags across the dataset.

- The first word cloud demonstrates the frequency of different tags across the entire dataset - we can see that JavaScript is the most popular tag.
- The second bubble plot isolates the tags that are programming languages. JavaScript, Java and Python lead the way as the most tagged programming languages in our dataset.
- The third word cloud focuses on the volume of duplicate questions per tag. We see a similar trend as observed in the first word cloud.

4.2 Feature extraction and analysis

We investigated the efficacy of a variety of features before selecting the five best suited to training the classifier. The FastText word embeddings did not produce sufficiently distinct distributions with any of the distance measures. Features like the absolute difference in question lengths, and ratio of question 1 to question 2 were not sufficiently complex for distinguishing duplicate and non-duplicate questions.

The most successful extracted features (SBERT and USE embeddings, common_words_ratio, sorted_fuzzy_ratio) had sufficiently distinct distributions with minimal overlap.

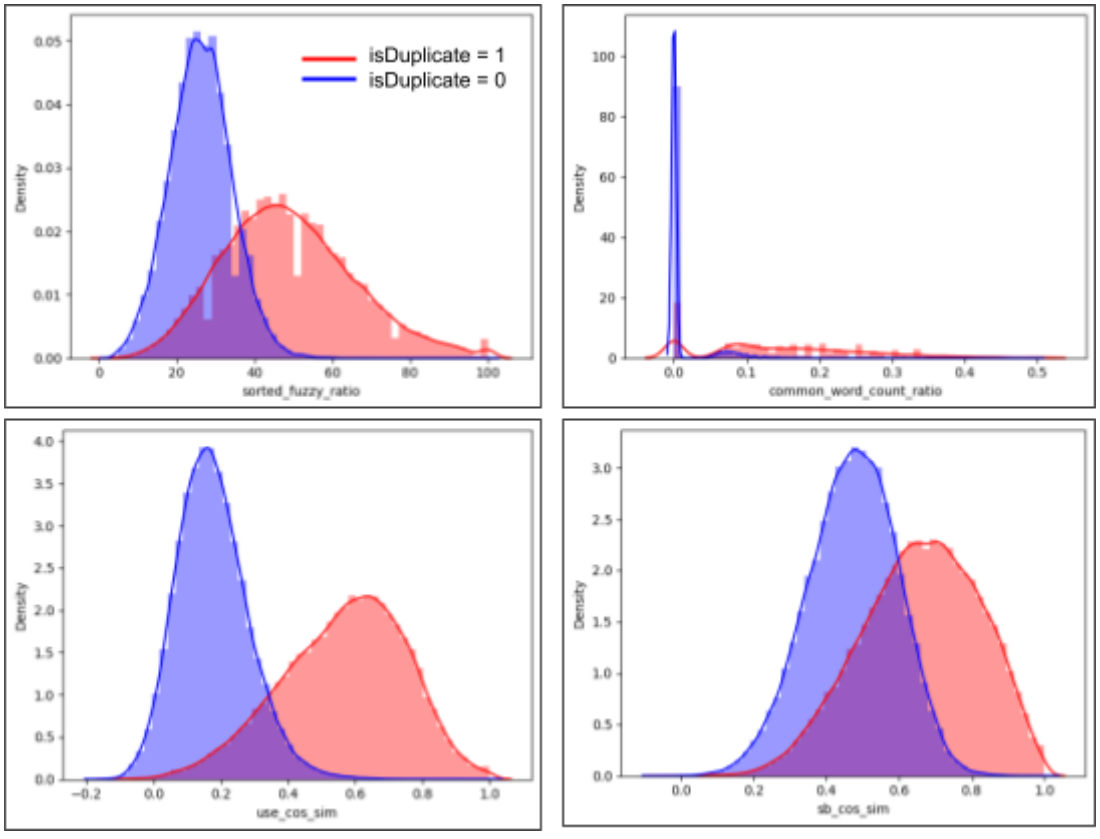


Figure 2 - Density distributions for some of the successful features used to train the classifier.

[1] This data was acquired by running a query on the StackExchange Data Explorer site: https://data.stackexchange.com/

Reducing the feature set down to a single dimension (as in the figure below) gives us a sense of the overlap in the distributions. The isDuplicate=1 and isDuplicate=0 distributions are sufficiently distinct for the purpose of classification.

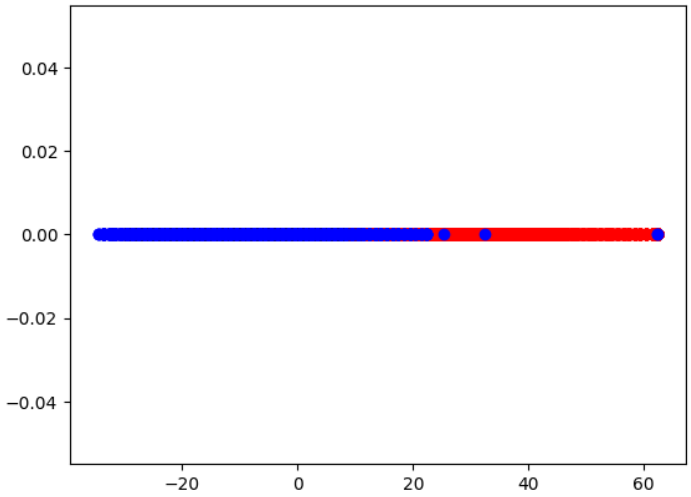


Figure 3 - PCA applied on feature set for classifier training

4.3 Question-pair similarity

The model was trained using the parameters discussed above and tested on a batch of data. The resulting confusion matrix demonstrates the accuracy of the model in determining question pair similarity.

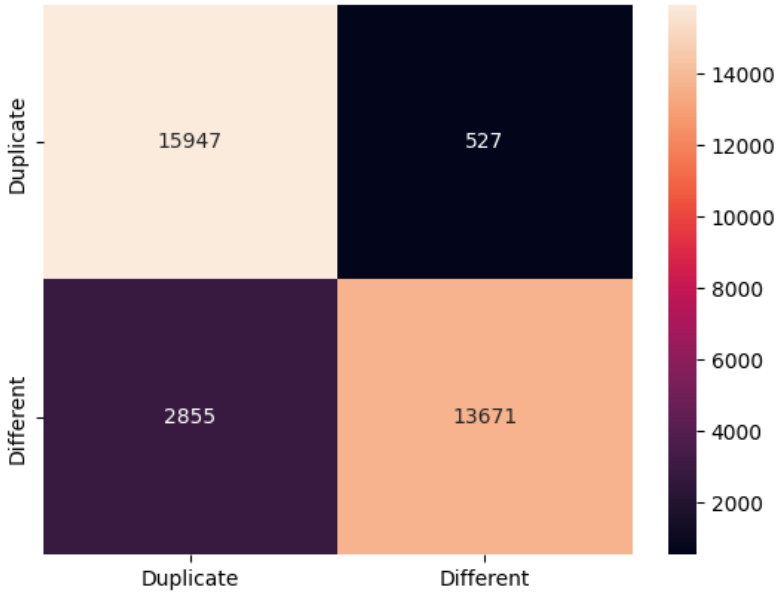


Figure 4 - Confusion matrix showing model accuracy on test data

The accuracy of the model: $ACC = \frac{TP + TN}{TP + FP + TN + FN} = \frac{15947 + 13671}{2855 + 527 + 15947 + 13671} = 0.905$

6 Summary

Category	Mark	Reasoning
Getting the data	1	Fetched the data from StackExchange data dump.
ETL	2	Queried, and structured the data using PySpark and Pandas.
Problem	1	Duplicate question analysis and inference on StackOverflow data
Algorithmic work	4	Feature engineering, embedding using SBERT and USE. Neural networks with Pytorch. Question search using Weviate vector space.
Bigness and parallelization	3	Using ORC file type to efficiently store and query data using PySpark. Reduced 80GB dump data to 20GB for analysis. Using ECR for storing the image for Docker and EC2 for scaling the app. Weviate can search large datasets under 100 ms.
UI	2.5	Used Streamlit to render front-end app.
Visualization	3.5	Using visualization tools (Tableau, Plotly) for visualizing data distributions. Used Seaborn for feature extraction and selection. Analysis of training data and full dataset using: time-series plots, density distributions, word clouds, etc.
Technologies	3	Python, PySpark, Pandas, Weviate, Tableau, ECR and ECS (AWS), Docker, Streamlit, Plotly, Pytorch, SBERT.
Total	20	