ResQue: Stack Overflow Question Recommendation System

Anindita Mishra
aninditamishra@tamu.edu
Texas A&M University

Kamala Akhila Mangipudi
akhila1012@tamu.edu
Texas A&M University

Varsha Khanna
vkhanna92@tamu.edu
Texas A&M University

Yi Liu
yl576@tamu.edu
Texas A&M University

ABSTRACT

Stack Overflow is a question and answer based platform for programming enthusiasts. It allows users to post questions to a large developer community and receive answers or advice about the issue the user is encountering. Often questions similar to the one being asked have already been answered and it might be useful for users to refer to such questions. We propose a system to recommend questions to users that are most relevant to the queried question and that have been previously answered. We propose to use Affinity Propagation algorithm to cluster similar questions together based on their vector representations. To generate recommendations for a given question we use cosine similarity against all other questions in the same cluster and suggest top n questions to the user. We also experiment with ways to incorporate additional features such as tag and answer information to improve our recommendations.

KEYWORDS

recommendation system, word embeddings, TF-IDF, clustering, cosine similarity

1 INTRODUCTION

The amount of resources and information available online has grown rapidly over the past few years. Online resources have become an invaluable tool for both understanding the problem as well as implementing the solution. StackOverflow is a Q & A forum that has gained prominence amongst the computing community for being one such resource. It is a flagship site of the Stack Exchange Network, which is a collection of Q&A websites on topics in diverse fields.

Stack Overflow is not only useful to the user posting a question but to any developer interested in gathering information. The aim of this project is to build a recommendation system for StackOverflow questions. Our recommendation system aims to help the user when:

- Questions similar to the one being asked have already been answered. These questions might have more relevant answers to the users information need or even provide the user with an alternative approach that they had not considered before.
- The user was not able to phrase his question effectively and therefore, was not able to find a solution for it.
- Getting a question correctly answered sometimes takes longer time and this wait time can be reduced if we find answers for contextually similar posts.

The proposed recommendation system has 5 stages. We begin by cleaning the available xml data to a more user friendly format. This was followed by representing the documents as their vector representations using pre-trained word embeddings. These are then fed to a clustering algorithm to group questions. For a given question, most similar questions within its cluster is determined using text similarity measures. We also experiment with different attributes associated with a question such as body, title and tags, and also use answer information to improve the performance of our recommendation system. Details regarding implementation and performance of our various models have been discussed in detail in the later sections.

2 RELATED WORK

There have been a few attempts made to develop a tool similar to the one proposed. In [7], they use variations of bag-of-words method to find similar questions using measures like Jaccard and cosine similarity. Similarity had to be computed against every single question in the corpus, each time, for any given question which made this approach computationally expensive. In [5], a modified Personalized PageRank algorithm was used to generate recommendations. This explored relations between Tags-Tags, Tags-Questions, Question-User-Tags as graphs. However, this algorithm was restricted to the nodes in the network formed and does not generalize to unseen queries. Therefore, we plan on addressing these two issues.

3 DATASET

We downloaded two data dumps provided by StackExchange[8] :Biology and Data Science. Each of these two datasets have the following schema :

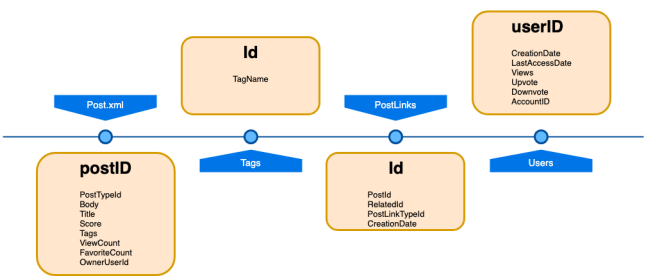

Figure 1: UML diagram of a sample data dump

- Posts.xml : has all the questions and answers title , body and Tag ids. "PostTypeId = 1" for questions and "PostTypeId = 2" for all the answers.
- PostLinks.xml : has related post Ids for each post which we are treating as our ground truth for evaluation.
- Tags.xml : contains tag Ids and their names.
- Users.xml : User profile information

We preprocessed these files to create corresponding questions, answers and postlinks CSV files.

Dataset & Questions & Answers
Biology & 21594 & 25111
Data Science & 14481 & 16785

Table 1: Total number of entries in each dataset

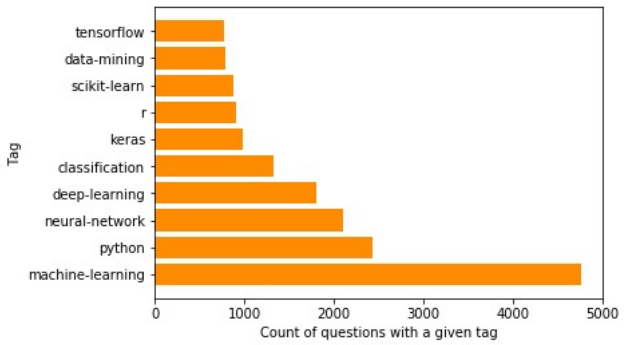

Figure 2: Keyword frequency distribution in question tags for Data Science dataset

4 METHODOLOGY

We employed two approaches to build our recommendation system: traditional TF-IDF model and a word embeddings based clustering model [Fig. 4]. Both of these models use cosine similarity measure to rank questions. In the traditional method, the whole whole corpus was used to rank similar questions. We intend to reduce the recommendation candidate pool size through clustering in our model. Finally, we also tested a hybrid method which combines both TF-IDF and embedding-clustering model to see if a better performance could be achieved.

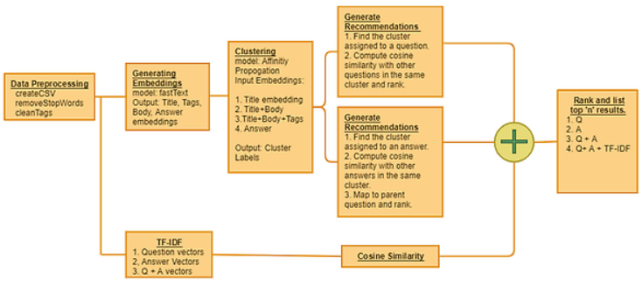

Figure 3: Diagram representing the Recommendation flow

4.1 Preprocessing

Raw data in XML files were parsed to remove HTML tags and extract desired data fields such question title, body and tag. The stop words were removed from the extracted text before tokenizing. This was finally stored into CSV files for later use.

4.2 TF-IDF

The result from TF-IDF method will serve as the baseline for our project. We created a TF-IDF matrix for questions and answers using TfidfVectorizer[9] from the scikit-learn library respectively. The question title, tag and body contents were used together to vectorize questions. Only answer body was used to vectorize answers. Pairwise cosine similarity for each question-question and answer-answer pair was calculated. For recommendations using answers, a list of similar answers was generated. The 'ParentID' of the answers was used to get the list of similar questions. Questions with the highest cosine scores were given as recommendations. Top 60 recommendations for each question Id were saved in a dictionary. Finally, question and answer contents combined and tested.

4.3 Embeddings

Embedding for questions and answers were done using a pre-trained word vector fastText model [2, 3]. The difference between fastText and word2vec is that fastText takes n-grams into consideration which is why we chose this library over word2vec. All texts including question title, body, tags and answer were split into words and vector representation for each word (300-dimension) was summed up and then averaged over number of words to get the representation for the specific entities (i.e., title, body, tags or answer).

To explore which part or parts provide the best representation for a question in terms of generating recommendations, we tested five different combinations: 1) title only; 2) title + body; 3) title + body + tags; 4) answer only; 5) question (title + body + tags) + answer. For each combination, the word embeddings were generated from the specified texts.

4.4 Clustering and Recommendation

We chose the Affinity Propagation [4] algorithm for clustering. Unlike K-Means [6], this algorithm does not require us to specify the number of clusters and therefore, can be easily extended to different datasets. Affinity Propagation algorithm operates by exchanging messages between pairs of samples until exemplars which are considered as most representative of other samples are chosen and convergence is reached [1]. A list of cluster centers were generated after running the algorithm and each question was labeled with a cluster ID.

After clustering is done, questions belonging to the same cluster as the query question and five nearby clusters were selected to generate the recommendation pool on which we use cosine similarity. Some clusters only have a small size of members and so we expand the candidate pool to nearby clusters. The nearby clusters were chosen by calculating Euclidean distances between cluster centers.

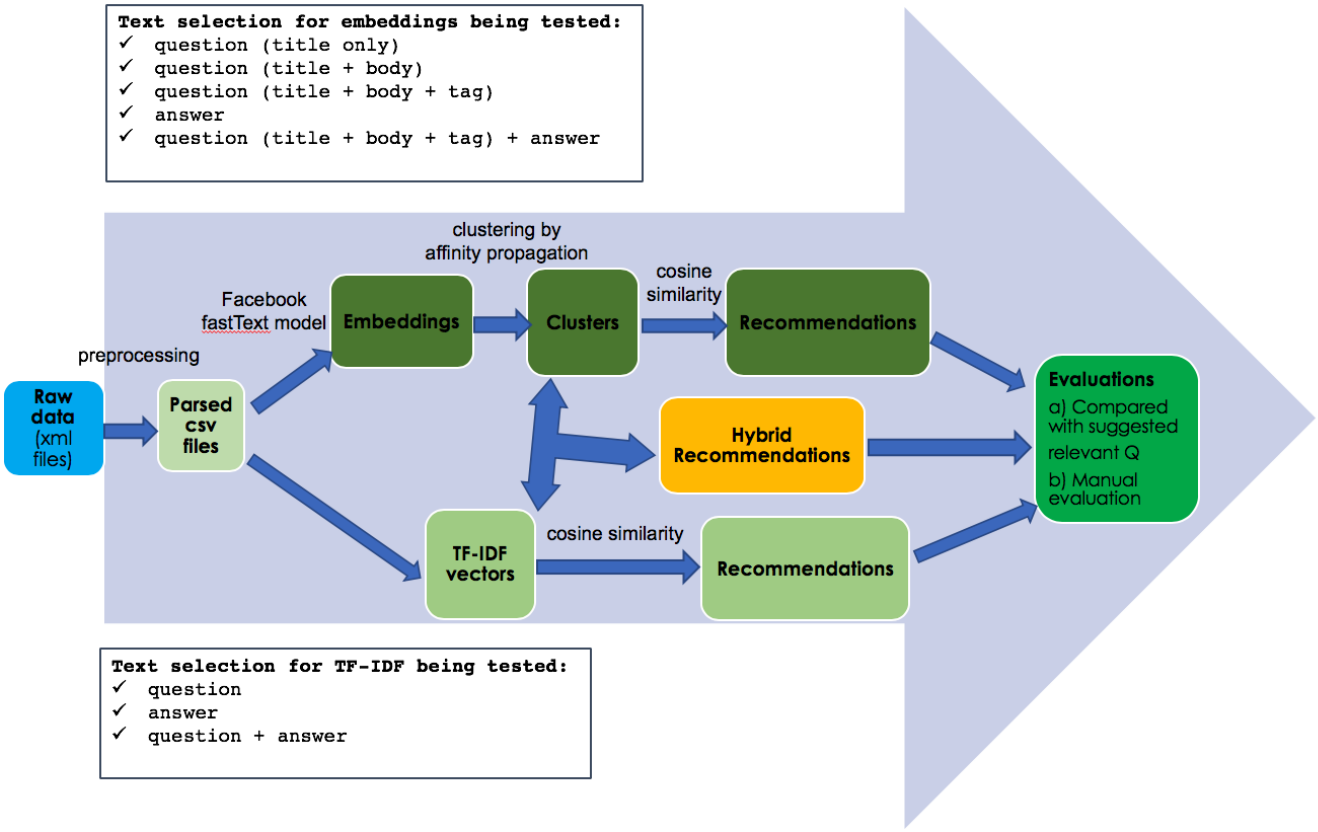

Figure 4: Model scheme for ResQue question recommendation system

4.5 Hybrid Method

Besides TF-IDF and embedding-clustering, we also tested a hybrid method by combining the recommendation results from both. Based on results from the five different varieties in embedding-clustering and three varieties in TF-IDF, a complete inclusion of question title, body, tag and answer contents to represent a question was deemed to be the best. So question title, body, tag and answer contents were all used in both TF-IDF and embedding-clustering, which are then used to calculate an equally weighted average of cosine similarity for the hybrid method.

4.6 Evaluation

PostLinks.xml file contained information regarding related questions for a question. We used this as ground truth to compare our recommendation results. A score of what percentage of our recommendation questions appeared in the related question list (true positive) was calculated and presented in the result tables [Table 2, Table 3]. This metric is similar to precision score. However, it also incorporates the rank of the recommended question. Relevant questions appearing in the top half of the recommendation list are given a higher weight.

Our evaluation metric depended greatly on the number and quality of related questions stated in the *PostLinks.xml* file. On inspection, we came across several examples where the provided list of related questions was either scarce or the given pool of related questions was not good enough to serve as ground truth. In order to better evaluate our system performance, we manually compared recommendation questions list with query question and judged whether they were related. Manual evaluation was conducted on the data science dataset. Forty random questions were selected and their recommendation results were evaluated [Table 2]. Manual evaluation was performed on the Data science dataset only and not the biology dataset.

5 RESULTS

As discussed in the methodology, we tested the performance of our model by using only questions, only answers and a combination of question and answer information.

Our baseline model is a TF-IDF model that uses the question title, body, tag and answers information. Our proposed method of using word embeddings has three variants where we include: only title information, only title and body information and title, body and tag information of the question.

We noticed that word embeddings did not represent domain specific keywords well whereas TF-IDF did. So, we decided to combine the two methods. The results for each of these methods on the Data Science and Biology datasets can be found in Table 1 and Table 2 respectively. The number (in %) in the tables is the modified precision metric used to evaluate the system as described in the evaluation subsection of methodology section.

We also made a GUI interface for users to interact with our recommendation system. A list of recommendation questions will be generated based on question title and body input using the hybrid method [Fig. 7 & 8].

Features	Attributes Used	Q	A	Q+A	Manual
TF-IDF	Title+Body+Tags	32.48	38.21	61.56	47.75
Embeddings	Title Only	7.42	7.03	9.5	
	Title+Body	7.39		11.08	
	Title+Body+Tags	13.04		13.25	40.25
Hybrid method	Title+Body+Tags	16.03	10.32	24.5	52.55

Table 2: Evaluation results on the Data Science dataset

Features	Attributes Used	Q	A	Q+A
TF-IDF	Title+Body+Tags	32.71	39.18	59.87
Embeddings	Title Only	9.29	13.4	18.4
	Title+Body	13.82		19.83
	Title+Body+Tags	18.9		22.9
Hybrid method	Title+Body+Tags	21.02	16.20	32.24

Table 3: Evaluation results on the Biology dataset

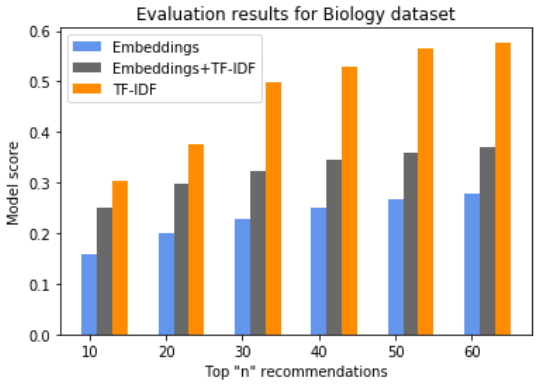

Figure 5: Evaluation of top-n recommendations for Biology dataset

6 CONCLUSIONS

- From the results it is seen that a combination of question and answer contents generates better recommendation results than using question only or answer only for both TF-IDF and embedding-clustering methods.

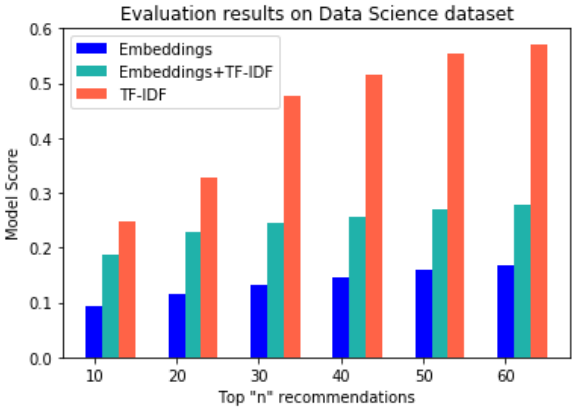

Figure 6: Evaluation of top-n recommendations for Data Science dataset

- The evaluation metric used can be heavily skewed due to the number related questions considered as ground truth is limited. Assuming our manual evaluation is consistent and trustful, TF-IDF's calculated score is close to the manual score while embedding and hybrid methods exhibit a large discrepancy between calculated and manual score.
- State-of-the-art word vector representation models do not cover highly specialized domain knowledge very well whereas TF-IDF is able to capture such technical terms. However, TF-IDF could not perform well for questions that were short and generic. By combining TF-IDF and embedding-clustering, our hybrid method is shown to perform better than either one of them. This performance boost is probably because the hybrid method gets the strength of both member methods.
- Questions that are more general have a larger pool of similar questions whereas specialized questions have a limited related question pool. This has an influence on the performance of our recommendation system. It performs better on general topic questions compared to specialized questions based on manual evaluation.
- It was observed that if the queried question belonged to multiple domains then the set of recommended questions were a mixture of suggestions from each of the prominent domains.
- The results of our recommendation system reinforced our intuition that if two answers are similar then their parent questions are also similar.

7 CHALLENGES

- Pre-trained word embeddings work well on general purpose datasets. However, these representations may not always transfer well to specialized datasets. Therefore, our system recommendations for specific questions were not as good when compared to simple or generic questions.
- The *PostLinks.xml* file consists of information regarding related posts for a given post. This was considered as the ground truth. However, less than 15% of questions present

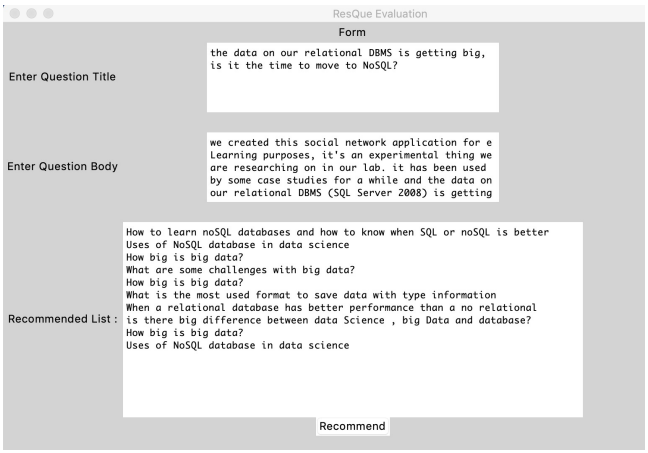

Figure 7: An example of a good recommendation using our system

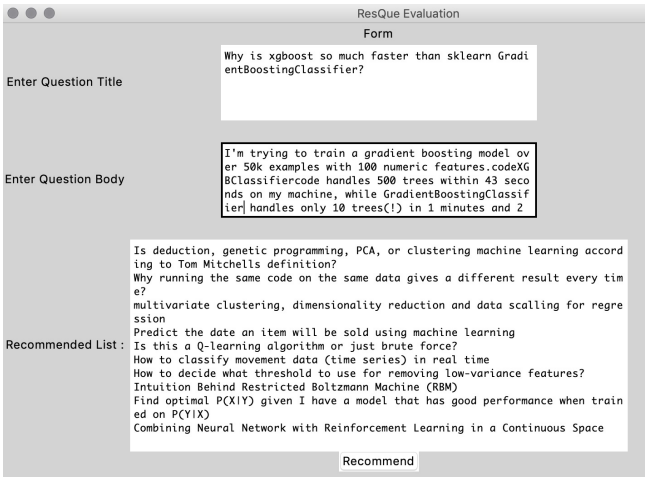

Figure 8: An example of a bad recommendation using our system

had related questions in the provided ground truth and each post only had less than three related posts. Therefore, it was difficult to find a good metric to evaluate the performance of our recommendation system.

- Manually evaluating our system was not a feasible solution due to it being a time consuming process and also due to lack of domain expertise to evaluate the results.
- Similar answers belong to questions that are similar. However, it was difficult to integrate answer information without adding noise and diluting question body signal.

8 FUTURE WORK

- Pre-trained word vectors that were trained using fastText were used to represent our data as vectors. However, this did not capture specialized domain vocabulary successfully. So as an extension, fastText or any other Word2vec model that is trained on our corpus of documents can be used to generate word embeddings. These embeddings would capture semantic relations of words in a domain more accurately.
- The data dumps provided by StackExchange have additional information like view count, favorite count and score associated with each post. These factors could be used to judge the reliability of a post.
- TF-IDF can be replaced by other BoW methods such Okapi BM25 that takes the length of documents into consideration while determining its relevance.

REFERENCES

[1] Affinity Propagation Algorithm. [n. d.]. https://scikit-learn.org/stable/modules/clustering.html#affinity-propagation.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[3] fastText. [n. d.]. https://fasttext.cc/docs/en/english-vectors.html.

[4] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[5] Jacob Perricone. [n. d.]. http://snap.stanford.edu/class/cs224w-2017/projects/cs224w-35-final.pdf.

[6] K-Means. [n. d.]. https://en.wikipedia.org/wiki/K-means_clustering.

[7] Shreya Bhatia, Arpita Sheth. [n. d.]. https://github.com/arsheth/DataScience/tree/master/Stack%20Overflow%20Recommendation%20System/.

[8] Stack Exchange Data Dump. [n. d.]. https://archive.org/download/stackexchange.

[9] TF-IDF Vectorizer. [n. d.]. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.