LINKING POSTS IN STACK OVERFLOW

Kiran Sai Gunisetty
Computer Science *1148912*
Lakehead University
Thunder Bay, Canada
kguniset@lakeheadu.ca

Venkata Naga Akshita Atmuri
Computer Science *1152600*
Lakehead University
Thunder Bay, Canada
vatmuri@lakeheadu.ca

***Abstract*—Stack Overflow posts make it easier to share information. The sheer number of questions and answers posted on the site clearly shows the popularity and success of this learning method. This also gave us a good source for researching the properties of posts. When searching for and resolving posts, linking posts to potentially linked posts, i.e., issue knowledge acquisition, will provide developers with more focused resources and information. However, since real-world acquiring is time-consuming, finding and acquiring similar posts is generally tricky and is dependent mainly on the individual developers' expertise and skills. As a result, automatically acquiring similar posts is a worthwhile activity that can improve development performance. The topic of gaining similar post information is formulated as a guideline problem in this article. We developed three models, TF-IDF and deep learning techniques, i.e., Word Embedding and Document Embedding, to solve this challenge. An analysis of the links, i.e., internal and external links, has also been performed.**

***Index Terms*—Stack Overflow, Unanswered Posts, Deep Learning, TF-IDF, Word embedding, Document embedding, Issue knowledge Acquisition;**

I. INTRODUCTION

Stack Overflow is a question-and-answer forum for experienced and enthusiast programmers. It is a privately-owned website founded in 2008 by Jeff Atwood and Joel Spolsky as the Stack Exchange Network flagship site. It contains questions and answers on a variety of programming languages and was developed as a more accessible alternative to previous question-and-answer platforms like Experts-Exchange. The forum acts as a medium for users to ask and answer questions and vote up or down questions and answers and update them in a similar way to a wiki or Reddit. Users of Stack Overflow can gain reputation points and "badges" for their valuable contributions; for example, an individual is given ten reputation points for having an "up" vote on a question or an answer to a question, and can earn badges for their valuable contributions, gamification of the conventional QA platform. With more reputation, users gain access to new features such as voting, commenting, and even editing other people's content. A sample stack overflow post can be seen in Fig 1.

The process of linking/acquiring similar posts, which we called issue knowledge acquisition, is essential for developers to quickly and efficiently resolve posts. We consider a pair of posts linked in our analysis if they have an association, such as dependent, equivalent, or referenced. Other developers will review and assess a new post after a user submitted it. During their conversation, other users can connect to similar posts to provide what they consider are valuable resources and information. The post would eventually be closed with the aid of relevant post information and discussions. Depending on the expertise and skills of individual developers, this manual acquisition process could take a long time. Thus, it is beneficial to provide automatic tools for acquiring similar posts in Stack Overflow.

Another similar platform is GitHub. GitHub incorporates the idea of social coding, allowing developers to network, cooperate, and support their projects in a developer-friendly world. Such social coding also contributes to increased code reuse as well as the problem resolution process. Developers can effectively contribute to topic reporting and

Fig. 1. Stack Overflow post

discussion. As a result, various developers often reported issues and addressed them at different times; it is not unusual to see that two issues contain similar details that can be exchanged to aid issue resolution. Many open issues in a project are connected to relevant issues through URL referencing during the topic conversation, one of the manifestations.

II. Related Work

One major issue in the online coding platforms is how to better coordinate the maintenance and expansion of shared resources. The owners and designers, in particular, are faced with a plethora of options for managing their massive and ever-changing populations. The institutional rules of a network may have an unpredictable impact on long-term outcomes. Indeed, the difficulty of these environments is shown by the fact that even minor site design decisions have been shown to have a significant impact on participation patterns. Sustainability and representativity are two major and interconnected challenges that these online communities face.

In [1], they suggest iLinker, a novel method for automatically acquiring related issue knowledge in GitHub projects. iLinker blends deep learning methods with conventional information retrieval techniques (TF-IDF) (word embedding and document embedding). They equate iLinker to a state-of-the-art methodology in conventional bug tracking systems called NextBug and other five baseline techniques, i.e., the mixture of their three sub-methods, in their analytical assessment.

Novice software developers often rely on code examples to learn how to use an API. Finding the characteristics of effective API examples is essential. To improve the appropriateness of these learning aids, it is necessary to identify the features of successful examples. In [2], the authors performed a comparative review of the questions and answers posted on Stack Overflow, a programming QA website, to help address this query. Answers on Stack Overflow may be voted on by users, showing which ones they find helpful. They discovered that the explanations that go along with the examples are almost as valuable as the examples themselves. Their studies have ramifications on how API documentation and examples can be created and evolved and the creation of resources that aid in the production of these materials.

In [3], they explore the importance of links in source code comments. They used a mixed-methods approach to classify the links' goals, aims, degradation, and evolutionary aspects in a large-scale analysis of about 9.6 million links to determine their prevalence. They discovered that links are widespread in source code libraries, that link goals include licenses, software homepages, and specs, and that links are often used to include metadata or attribution. While links are occasionally modified, many link targets do. Almost 10% of the connections in source code comments are no longer active. They then sent a slew of link-fixing pull requests to open-source software repositories, with the majority of their patches successfully integrated. Their results suggest that links in source code comments may be weak, and their work paves the way for further research into these issues.

In [4], they look at major issues on Stack Overflow, the reasons that could be related to new contributors' move from posing questions to posting responses. They discovered that a user's personal characteristics, such as tenure, gender, geographic area, and characteristics of the sub-community in which they are most involved, such as the scale and frequency of negative social reviews, have

an enormous impact on their willingness to post responses. Their paper takes a first look at the barriers and threats to online user contributions by assessing and modeling these relationships.

In [5], the authors look at the mutual information sharing between Android Issue Tracker and Stack Overflow as an example. Centered on the internal citation graph, they suggest an automated approach that combines semantic similarities with temporal locality between Android issues and Stack Overflow posts. Their method looks at internal citations in forums to see if there are any closely associated posts or problems. It then uses the temporal similarities between issues and posts to rate associations. Extensive testing shows that their method has a precision of 62.51 percent for top 10 suggestions when recommending Stack Overflow posts to Android issues and a precision of 66.83 percent when recommending Stack Overflow posts to android issues.

III. Dataset

We have taken the data from the following Stack Exchange link. It contains data of all stack exchange websites. So we downloaded the Stack overflow related data alone for our project. It can be seen as shown in Fig 2. Once we have downloaded the datasets, we used the eTree parse to convert the XML files to CSV files. The main tables used are **Posts, Comments and PostLinks**.

IV. Methodology

A. *RQ-1:*

Analysis of internal and external links in stack overflow posts (questions, comments, and answers)

The first research question is divided into three parts-

- The first one is about analyzing the percentage of links (both internal and external) in Stack Overflow posts. The results are plotted in a pie chart. As we can see in Fig 3., most percentage of the links are present in Answers (58.5%) followed by Comments (22%) and Questions (19.6%). It is understandable because users while answering the posts often refer to other links which may be internal to stack overflow or some other external websites .

Fig. 2. Dataset link

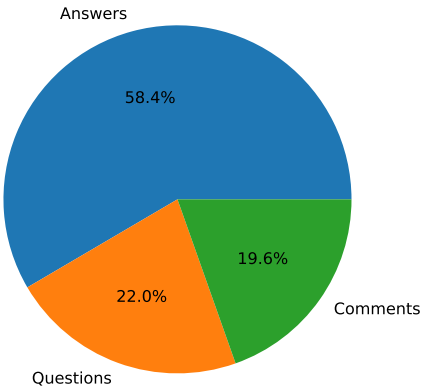

Fig. 3. Percentage of Links in SO Posts

- The second one is about the top domains in external links. We have collected all the ex-

ternal links in Questions, Answers, and Comments. Then using the 'urllib' library, we have extracted domains from the text input. Then top twenty domains have been filtered out and projected in a bar graph. As shown in Fig 4., the topmost domains were the Microsoft and ASP domains, whereas the least number of links were among MySQL and Codeproject domains. Microsoft domain provides documentation to a lot of other softwares such as Visual studio, SQL server, Azure and a lot more. It may be the reason for more number of references in stack overflow posts.

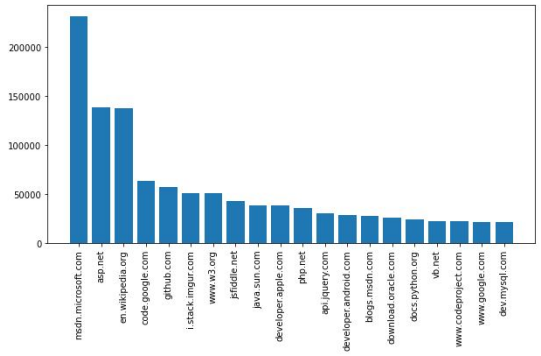

Fig. 4. Top 20 External Links

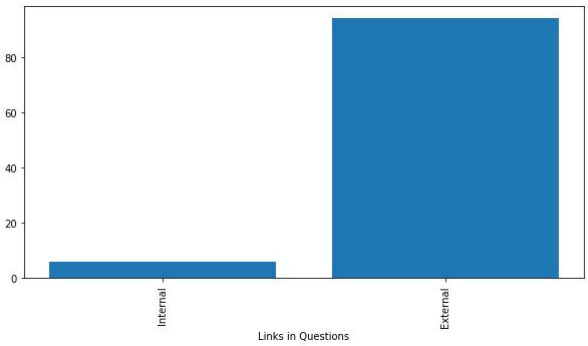

Fig. 5. Percentage of Links in Questions

- The third one is about the percentage of internal and external links in Questions, Answers, and Comments. We extracted the links from each post and categorized them into 2 categories- internal and external. Then we plotted 3 bar charts for each category, i.e., Questions, Answers, and Comments. The following figures (Fig 5, Fig 6, Fig 7) shows the output. It can be seen that externals links are more in stack overflow posts compared to the internal posts.

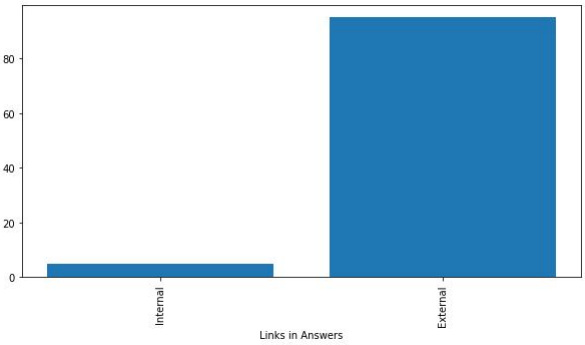

Fig. 6. Percentage of Links in Answers

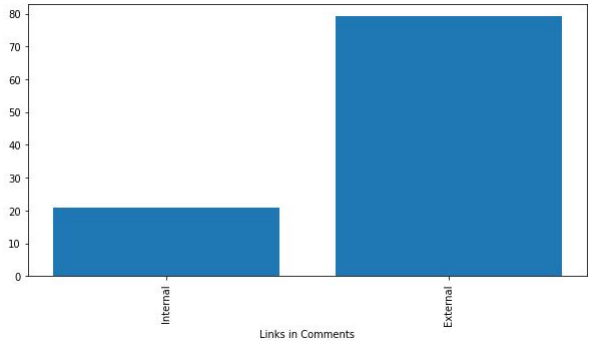

Fig. 7. Percentage of Links in Comments

B. RQ-2:

Analysis of time taken for linking related posts in stack overflow.

The second research question is divided into two parts-

- We have calculated the percentage of links acquired within a day for each post in the first part. We then divided the output into the

following five categories -

- < one_hr
- 1-3 hrs
- 3-6 hrs
- 6-12hrs
- > 12hrs

The output is plotted in a bar graph as shown in Fig 8. From the image, we can infer that most of the links for posts are acquired within 1 hr, followed by the “more than 12 hrs category”. On the other hand, the least percentage of links acquired was between the ”6 to 12 hrs category”.

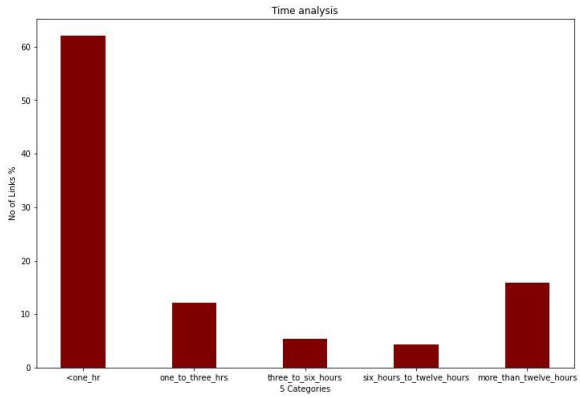

Fig. 8. Percentage of Links acquired Part- 1

- In the second part of the question, we have calculated the percentage of links acquired in the whole timeline of each post. We then divided the output into the following five categories -

  - < one_day
  - 1-7 days
  - 7 days to 2 weeks
  - 2 weeks to 4 weeks
  - > 1 month

Here also, most links are acquired within 1 day followed by 1-7 days category as seen in Fig 9. We have excluded the <one_day category to show the remaining categories better, as shown in Fig 10.

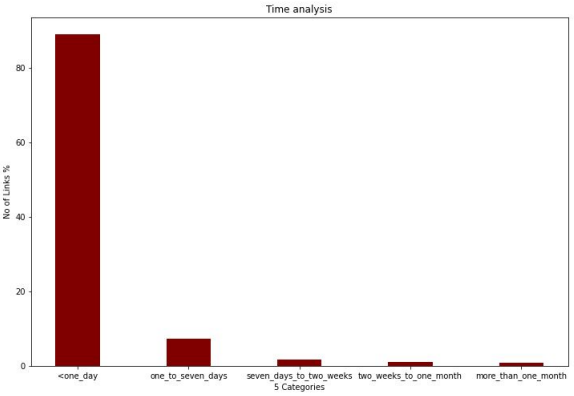

Fig. 9. Percentage of Links acquired Part- 2a

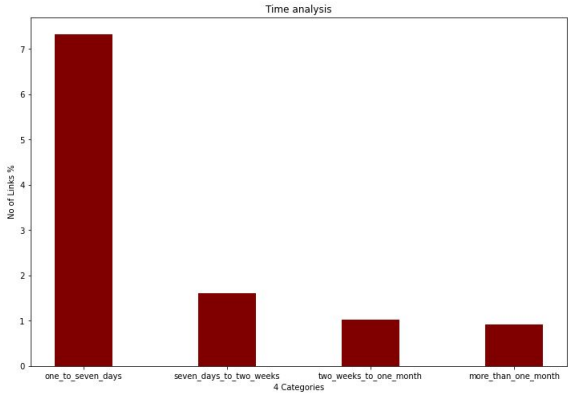

Fig. 10. Percentage of Links acquired Part- 2b

C. *RQ-3:*

Building a model that returns the topmost related posts for the unanswered questions in SO.

The basic flow of our work done for the third research question can be seen in Fig 11.

- The third research question focused on developing models that would recommend related posts to unanswered questions on stack overflow. For training and testing our

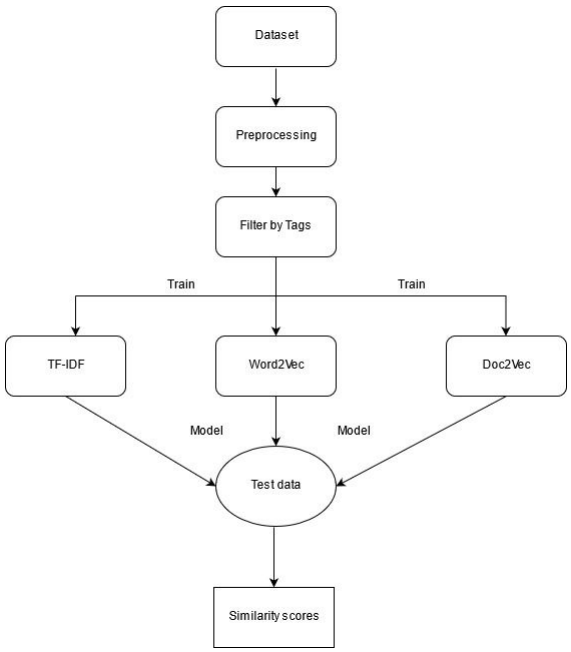

Fig. 11. Architecture

models, we used questions related to the python tag specifically. It is because it has the fastest-growing percentage over the years, as we see in Fig 12., besides that, it also reduces the data to be trained, and we can see how well the model can perform for a single tag and later we can implement it for all.

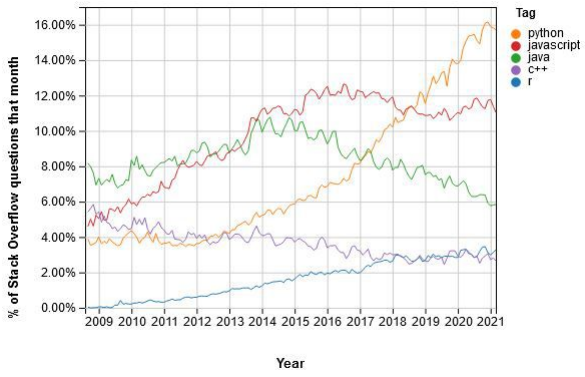

Fig. 12. Trends

- The three models developed were -
  - **TF-IDF:** TF-IDF stands for Term frequency-inverse document frequency. used in information retrieval systems and also content-based filtering mechanisms (such as a content-based recommender).
  - **WORD2VEC:** The word2vec algorithm learns word associations from a vast corpus of text using a neural network interface. Once learned, a model like this can identify synonyms and recommend alternate terms for a sentence. The Continuous Bag-of-Words model and the Skip-gram model are the two main models described in word2vec.
  - **DOC2VEC:** Doc2vec is an NLP tool which creates a numeric representation of a document. It represents documents as a vector and is a generalization of the word2vec method. PV-DM and PV-DBOW are two main methods in Doc2Vec.

Parameters used in TF-IDF [6]:

- **analyzer:** This parameter is used to choose whether the feature should be made of word or character n-grams.
- **ngram_range (tuple (min_n, max_n), default=(1, 1)):** This parameter decides the lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n <= n <= max_n will be used. For example, a ngram_range of (1, 1) means only unigrams, and (2, 2) means only bigrams. It only applies if the analyzer is not callable.
- **min_df (float or int, default=1):** When building the vocabulary, ignore terms with a strictly lower document frequency than the given threshold. If float in the range of [0.0, 1.0], the parameter represents a proportion of documents, integer absolute

counts.
- **stop_words:** A string is passed to _check_stop_list and the appropriate stop list is returned. English is currently the only supported string value. There are several known issues with English, and you should consider an alternative.

Parameters used in Word2Vec [7]:

- **size/vector_size (int, optional):** Dimensionality of the word vectors.
- **window (int, optional):** Maximum distance between the current and predicted word within a sentence.
- **min_count (int, optional):** Ignores all words with total frequency lower than this.
- **workers (int, optional):** Uses these many worker threads to train the model (=faster training with multicore machines).
- **sg (0, 1, optional):** The first involves predicting context terms using a center phrase, while the second involves predicting the word using context words. The skip-gram model was used in our project. The SkipGram model learns to infer a term from its surroundings. Even though it takes more time for training than CBOW, it works well with a small amount of the training data and represents well even rare words or phrases.

Parameters used in Doc2Vec [8]:

- **documents (iterable of list of TaggedDocument, optional):** Input corpus can be simply a list of elements, but for larger corpora, consider an iterable that streams the documents directly from disk/network.
- **dm (1,0, optional):** Defines the training algorithm. If dm=1, 'distributed memory' (PV-DM) is used. Otherwise, a distributed bag of words (PV-DBOW) is employed.
- **vector_size (int, optional):** Dimensionality of the feature vectors.
- **window (int, optional):** The maximum distance between the current and predicted word within a sentence can be chosen using this parameter.
- **alpha (float, optional):** To decide the initial learning rate of the model.
- **min_count (int, optional):** The model will ignore all words with total frequency lower than this.
- **sample (float, optional):** This is used to set the threshold for configuring which higher-frequency words are randomly downsampled, useful range is (0, 1e-5).
- **workers (int, optional):** Use these many worker threads to train the model (=faster training with multicore machines).
- **dm_concat (1,0, optional):** If 1, use concatenation of context vectors rather than sum/average; Note that concatenation results in a much larger model, as the input is no longer the size of one (sampled or arithmetically combined) word vector, but the size of the tag(s) and all words in the context strung together.
- **dbow_words (1,0, optional):** If this is set to 1, then training is done using word-vectors (in skip-gram fashion) simultaneous with DBOW doc-vector training; If 0, only trains doc-vectors (faster).

The Title and Body of each post have been taken and combined into a new column that is pre-processed. In pre-processing, we have removed the punctuations, special characters, and stop words. Then the models have been trained with the pre-processed data.

V. CODE

Programming language which we used to code is Python. Code comprises of some important libraries such as scikitlearn, pandas, numpy and gensim. The parameters used have already been mentioned in the Methodology section. The project is divided into three files, one for each research question - Link analysis in Posts and Comments, Time analysis for Linking Posts and Recommendation. We have uploaded this code in GitHub and shared

the repository link in the references [9].

VI. Results

We have chosen PostId and Title of ten unanswered posts (shown in Fig 13) from the stack overflow dataset. We have run the models for each post in that list to get the 10 topmost related PostIDs along with the similarity scores for each. Then we collected the highest similarity score from the 10 PostIDs recommended for each post. We tuned the models with different hyperparameters to get the best output. The output of the three models can be seen in Fig 14,15,16.

```
337 - XML Processing in Python
469 - How can I find the full path to a font from its display name on a Mac?
502 - Get a preview JPEG of a PDF on Windows?
535 - Continuous Integration System for a Python Codebase
594 - cx_Oracle: How do I iterate over a result set?
683 - Using 'in' to match an attribute of Python objects in an array
742 - Class views in Django
766 - Python and MySQL
773 - How do I use itertools.groupby()?
972 - Adding a Method to an Existing Object Instance
1171 - What is the most efficient graph data structure in Python?
```

Fig. 13. Unanswered Posts

We can see that Doc2Vec similarity scores are higher compared to the other two. The scores were ranging between 0.65 and 0.78. On the other hand, similarity scores of TF-IDF ranged between 0.39 and 0.56, and Word2Vec similarity scores ranged between 0.49 and 0.74. TF-IDF has several disadvantages compared to word embedding and doc embedding techniques, and this might be the reason for its lower similarity scores. A few of the drawbacks of using the TF-IDF model are :

- It computes document similarity directly in the word-count space, which may be slow for large vocabularies.
- It assumes that the counts of different words provide independent evidence of similarity.
- It makes no use of semantic similarities between words.

Fig 17 shows the table comparing the similarity scores of all models.

```
Highest score for Post Id 337 is = 0.5602681983303459
Highest score for Post Id 469 is = 0.47302601673938777
Highest score for Post Id 502 is = 0.41127725075008614
Highest score for Post Id 535 is = 0.4647230425239056
Highest score for Post Id 594 is = 0.42707757837824106
Highest score for Post Id 683 is = 0.4119239741093581
Highest score for Post Id 742 is = 0.3749356990365732
Highest score for Post Id 766 is = 0.47424417298691657
Highest score for Post Id 773 is = 0.39268621842216617
Highest score for Post Id 972 is = 0.5063109556579887
Highest score for Post Id 1171 is = 0.5027743386644267
```

Fig. 14. TF-IDF Output

```
Highest score for Post Id 337 is = 0.5556448101997375
Highest score for Post Id 469 is = 0.5496886968612671
Highest score for Post Id 502 is = 0.6351563930511475
Highest score for Post Id 535 is = 0.5437466502189636
Highest score for Post Id 594 is = 0.7417551875114441
Highest score for Post Id 683 is = 0.49396514892578125
Highest score for Post Id 742 is = 0.5088351964950562
Highest score for Post Id 766 is = 0.5553408265113831
Highest score for Post Id 773 is = 0.5059433579444885
Highest score for Post Id 972 is = 0.6138319969177246
Highest score for Post Id 1171 is = 0.5723389387130737
```

Fig. 15. Word2Vec Output

```
Highest score for Post Id 337 is = 0.6675436496734619
Highest score for Post Id 469 is = 0.6557201147079468
Highest score for Post Id 502 is = 0.7186626195907593
Highest score for Post Id 535 is = 0.6886473894119263
Highest score for Post Id 594 is = 0.7548345327377319
Highest score for Post Id 683 is = 0.6817964315414429
Highest score for Post Id 742 is = 0.6681376695632935
Highest score for Post Id 766 is = 0.6960107088088989
Highest score for Post Id 773 is = 0.6520678400993347
Highest score for Post Id 972 is = 0.7824229001998901
Highest score for Post Id 1171 is = 0.6779126524925232
```

Fig. 16. Doc2Vec Output

Post ID	TF-IDF	Word2Vec	Doc2Vec
337	0.5602	0.5556	0.6675
469	0.473	0.5496	0.6557
502	0.4112	0.6351	0.7186
535	0.4647	0.5437	0.6886
594	0.427	0.7417	0.7548
683	0.4119	0.4939	0.6817
742	0.3749	0.5088	0.6681
766	0.4742	0.5553	0.696
773	0.3926	0.5059	0.652
972	0.5063	0.6138	0.7824
1171	0.5027	0.5723	0.6779

Fig. 17. Comparison

VII. Limitations

Our criteria for building our sample was too restrictive as we focused on only Python tag related posts. More posts related to different tags may increase the consistency of the results, but it is known that different posts have different characteristics, which may cause some bias. In our preprocessing of issue documents, we used WordNetLemmatizer for transforming the words to their root forms. In future work, we plan to use different stemmers (such as Lancaster stemmer) for this purpose and compare their performances with WordNetLemmatizer. Another concern would be impact of duplicate questions in the training set.

VIII. Future work

From our study results, we can still do a rich resource of ideas for future research.

- **Future work should further investigate the impact of duplicate questions in training set upon the performance of the model**. In our current study, we haven't separated the duplicate questions from the dataset that was used for training. So one can use the PostLinks dataset to filter out the duplicate questions from the dataset and train the model to check if it is performing better.
- **Future work should include the analysis of the unanswered questions**. It may be possible that the question may be incomplete or not having accurate information about the problem. Why do questions at Stack Overflow remain unresolved for a long time?
- **Future work should look forward to implementing a pulg-in for stack overflow website which will recommend related posts immediately once a new question has been posted** . As we can see from the results of the second research question, some posts take more time to get related links. So it would be helpful for the developers if there's a plug-in kind of thing that will recommend a set of related posts as soon as a question has been posted or while posting a question.

IX. Conclusion

In our study, we have analyzed links in stack overflow posts. We explore three different research questions. We have used the traditional information retrieval technique TF-IDF and deep learning techniques word embedding and document embedding. In our evaluation, we can see that Doc2Vec performs better compared to the other two techniques. Furthermore, we find that tag-specific training corpus can improve the performance of deep learning techniques, and different training and testing sets have negligible effects on the performance. Our future work will further enhance the approach by considering more information about lexical, semantic, user behavior, and popularity associated with the questions.

References

[1] Yang Zhang , YiwenWu , Tao Wang , Huaimin Wang, **"iLinker: a novel approach for issue knowledge acquisition in GitHub projects"**, WorldWideWeb (2020) 23:1589–1619.

[2] Seyed Mehdi Nasehi, Jonathan Sillito , Frank Maurer, and Chris Burns, **"What makes a Good example? A Study of Programming QA in StackOverflow "**, 2012 28th IEEE International Conference on Software Maintenance (ICSM).

[3] Hideaki Hata, Christoph Treude, Raula Gaikovina Kula and Takashi Ishio, **"9.6 Million Links in Source Code Comments:Purpose, Evolution, and Decay"**, 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE).

[4] Timur Bachschia, Anik o Hann akb, Florian Lemmericha, and Johannes Wachs, **"From Asking to Answering : Getting More Involved on Stack Overflow"**, arXiv:2010.04025v1 [cs.CY] 8 Oct 2020

[5] Huaimin Wang, Tao Wang, Gang Yin, and Cheng Yang, **"Linking Issue Tracker with QA Sites forKnowledge Sharing across Communities"**, IEEE Transactions on Services Computing, VOL. 11, NO. 5, September/October 2018

[6] **"https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html"**

[7] **"https://radimrehurek.com/gensim/models/word2vec.html"**

[8] **"https://radimrehurek.com/gensim/models/doc2vec.html"**

[9] Kiran Sai Gunisetty, Venkata Naga Akshita, 2021, Linking Posts in Stack Overflow, **"https://github.com/kinnu183/linking-posts-stack-overflow.git"**