Mining Stack Overflow: a Recommender Systems-Based Model

Fouzi Harrag
Computer Sciences Department,
College of Sciences,
Ferhat Abbas University,
Setif, Algeria,
fouzi.harrag@univ-setif.dz

Mokdad Khamliche
Computer Sciences Department,
College of Sciences,
Ferhat Abbas University,
Setif, Algeria,
khemliche.mokdad@gmail.com

Abstract – In software development, developers received bug reports that describe the software bug. Developers find the cause of bug through reviewing the code and reproducing the abnormal behavior that can be considered as tedious and time-consuming processes. The developers need an automated system that incorporates large domain knowledge and recommends a solution for those bugs to ease on developers rather than spending more manual efforts to fixing the bugs or waiting on Q&A websites for other users to reply to them. Stack Overflow is a popular question-answer site that is focusing on programming issues, thus we can benefit knowledge available in this rich platform. This paper, presents a survey covering the methods in the field of mining software repositories. We propose an architecture to build a recommender System using the learning to rank approach. Deep learning is used to construct a model that solve the problem of learning to rank using stack overflow data. Text mining techniques were invested to extract, evaluate and recommend the answers that have the best relevance with the solution of this bug report.

Keywords – Recommender System, learning to rank, Mining software repositories, Text Mining, Deep learning, Stack Overflow.

1. INTRODUCTION

In software development area, there is huge amount of unstructured data that grows fastly every day. This data exists in different levels and systems used in the software development process such as versioning systems, issue trackers, achieved communications systems and many other repositories. Mining the rich software engineering data represents a modern field for data mining domain that will open big doors for researchers and developers. In fact, the investigating in such data will make revolution in development and maintenance activities.

This paper represent a state of the art for this research topic. Section 2 of this paper covers the definitions and the overview of software repositories and describe as well each type of these repositories. We will also give a global presentation of Stack overflow platform considered in our study case as a knowledge repository.

Section 3 summarizes the state of modern mining software repositories area and gives definition of data mining, recommender System, NLP techniques. Section 4 presents an overview of related works in the field of mining software repositories with discussion of different proposed approaches. Section 5 is dedicated to the presentation of our proposed approach. We focus the concepts and techniques of machine and deep learning that will be considered in the construction of our framework.

2. Software data

Unstructured data refer to information that is not organized by following a precise schema or structure. Such data often include text (e.g., email messages, software documentation, and code comments) and multimedia (e.g., video tutorials, presentations) contents. These kinds of data are estimated to represent 80% of the overall information created and used by enterprises in software projects [1]. Large amount of artefacts are generated in development process. This huge data is continuously growing over time. It is called software repositories.

2.1. Software repositories

Ahmed E. Hassan define software repositories [1] as a record-keeping database that stores data about artifacts of a complex computer based system. It tracks changes applied to the artifact and stores corresponding Meta data. Source control repositories, bug repositories, archived communications, deployment logs, and code repositories are examples of software repositories that are commonly available for most software projects.

2.1.1 Type of software repositories

Through the type of information and their purpose in this repositories. We can divide them to the following types:

- **Historical repositories**: record information about the evolution and progress of project. This type of repositories can contain tracks of all changes of the source code, the historical bug reports, the communication between team of development Such as: Version control systems (*CVS, SVN, Git, Mercurial*), Bug repositories (*Bugzilla, JIRA*), Mailing lists (*e-mails, wiki pages*) and Development collaboration sites (*Stack Overflow*).

- **Code repositories**: contain source code of various applications Developed by several teams of developments Such as Code bases (*Source Forge, Google Code*) and Project ecosystems (*GitHub*).

- **Runtime repositories**: contain information about the execution and usage of an application Such as Crash reports, Field logs, and Execution traces.

- **Mobile Application repositories:** contains the logs and bug report of applications mobile, feedbacks of users Such as: App Stores (*Google Play Store, Apple App Store*), mobile apps user feedbacks (reviews, ratings).

Figure 1 shows examples of the current and historical artifact and interaction that are registered in software repositories.

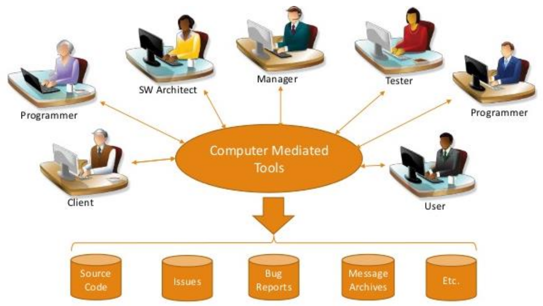

Figure 1 Types of Software Repositories [2]

2.1.2 Importance of Software repositories:

➢ Software repositories contain a wealth of valuable information about software projects, that's why they are considered as a first class source of information.
➢ Software engineering becomes as a more and more data-driven discipline lowering the dependency on intuition and experience of developers.
➢ When we applies mining techniques on software repositories, we win a lot of time and costs and we will increase the productivities of development and maintenance software projects.

2.1.3 Stack Overflow as a Knowledge Repository

Social media, unlike traditional media, gives people an easy way to communicate, collaborate and share information with each other's. There are many types of social media such as blogs, micro blogs, bookmarking sites, social news, media sharing and social networks. In the present time, Social media turns into social productivity through the participation of individuals in their ideas, experiences and skills to generate content. This collective work is called crowd sourcing. D.Brayvold [3] defined crowd sourcing as "the process of getting work or funding, usually online, from a crowd of people". It generates content by participation of a large group of people in their skills, ideas, and knowledge. The usage of crowd sourcing can help companies and individual in doing their tasks with low cost compared to employing specialists as well providing a high number of people who are ready to work anytime. One of the most important crowd sourcing platforms is stack overflow.

Stack overflow depends on the crowd to provide accumulated and to construct quality developer-related knowledge. Extracting knowledge from Stack Overflow by using data mining will be a great success of major interest for the community. In the next section, we will outline and clarify the concept of mining software repositories.

2.1.4 What is Stack overflow?

Ahasanuzzaman et al [4] define Stack Overflow as "a popular question answering site that is focused on programming problems». In this community, Users can ask questions, provide answers to the questions asked, mark the questions as favorite, up vote / down vote an answer, tag questions, and carry out other community related tasks. Programmers have actively used it to ask questions from January 2009. Today there is more than 18 million questions, 27 million answers, and 10 million users.

In addition to the above, Stack Overflow makes some mechanisms of tagging and earning reputation, badges to have more privileges in the community. These mechanisms are explained in the following lines:

Tags: Stack Overflow allows users to tag each question, with up to a maximum of five tags. Users can select an existing tag provided in the autocomplete text box or create a new one. To create a new tag, users need to have a minimum level of reputation on Stack Overflow. This makes sure that only expert users can create new tags, which is maintaining consistency among tags found on Stack Overflow. Expert users can also change the question tags, in case they found them incorrectly tagged.

User Reputation: Stack Overflow provides a metric called *Reputation* to rank their users. Reputation is an approximate measurement of how much the community trusts a user; it is earned when the peers appreciate what a user is contributing. Users do not need reputation for basic site functionalities such as asking questions and providing answers, however users with high reputation score gain more privileges. The primary way to gain reputation is by posting good questions and useful answers. Votes on these posts cause user to gain (or sometimes lose) reputation. The maximum number reputation points that can be earned in a day is 200, thus making sure that the reputation gained by a user is by actively and consistently participating in the site activities.

Privileges: are the accesses that user unlock each time it win a certain reputation. For example, user can negatively vote a position only if it has at least 15 reputations. The last privilege is that which will allow user to have access to the Google analytics data of the site.

Badges: user can earn badges when it perform a specific predefined operation. For example, earning the Supporter badge when user vote positively for the first time.

3. Mining Software repositories (MSR)

The field of mining software repositories aims at examining and analyzing “the rich data available in software repositories to uncover interesting and actionable information in about software projects and systems” [5].

3.1. Application of MSR

- **Prediction and identifying bugs**: Predicting the occurrence of bugs remains one of the most active areas in software engineering research. By using MSR, it is possible to predict and localize the bugs, so managers can allocate testing resource appropriately, developers can review risky code more closely, and testers can prioritize their testing efforts.

- **Understanding Software Systems**: Understanding large software systems remains a challenge for most software organizations. Documentations for large systems rarely exist and if they exist, they are often not up-to-date. Information stored in historical software repositories, such as mailing lists and bug repositories, represent a group memory for a project. Such information is very valuable for current members of a project.

- **Understanding Team Dynamics:** Many large projects communicate through mailing lists, IRC channels, or instant messaging. These discussions over many important topics such as plans, design decisions, project policies, and code or patch reviews. These discussions represent a rich source of historical information about the inner workings of large projects. The mining of these discussions can help better understand the dynamics of large software development teams.

- **Propagating Changes**: Change propagation is the process of propagating code changes to other entities of a software system to ensure the consistency of assumptions in the system after changing an entity. For example, a change to an interface may require the change to propagate to all the components, which use that interface. Instead of using traditional dependency graphs to propagate changes, we could make use of the historical co-changes. The intuition is that entities co-changing frequently in the past are very likely to co-change in the future.

3.2. Data mining for Software Engineering

Data mining is the science of extracting useful knowledge from such huge data repositories, to use this knowledge in the decision process [6]. Data mining is aims to discover hidden and useful patterns in huge data sets. Data Mining is all about discovering unsuspected and unknown relationships amongst the data. Data mining uses machine learning, statistics, AI and database technology to provide reliable results.

3.2.1 Machine Learning

Tom Mitchell in his book Machine Learning [7] provides another definition: “The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience”. Like Humans have the ability to learn by experience. Machines with machine learning can do the same. The goal of machine learning processes is to generate in output a predictive Model based on data used in training. Depending on the nature of the business problem being addressed, there are different approaches based on the type and volume of the data: supervised learning, unsupervised learning and reinforcement learning [8].

3.2.2 Deep Learning

Deep learning focuses on a specific category of machine learning called Artificial Neural Networks that is inspired from the functionality of the human brain. Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training example [9].

3.2.3 NLP Techniques for Data Preprocessing

NLP is the capacity of a computer program to comprehend human language [10]. Researchers use NLP techniques to preprocess data before applying IR models to the data. Preprocessing steps include tokenization, splitting, stop word removal, stemming and pruning. IR overlaps with other fields, especially database technology and natural language processing (NLP). Information Retrieval techniques and algorithms are also used in the recommender systems research field.

4. Related Works

Many researchers have discussed the effectiveness of using data mining techniques to facilitate the debugging process for software engineering developers. This section presents an overview of the state of the art of this research area.

Xin et al [11] presented a ranking approach that simulates the bug locating process used by developers. The ranking model benefited from domain knowledge such as API specifications, the bug-fix history, and code-change history. "The ranking score of each source file is calculated as a weighted combination of an array of features ". Evaluation of experimental results was done on six open sources java projects which are Eclipse, JDT, Birt ,SWT. Tomcat and AspectJ. Results showed that the ranking approach is better than BugLocator, VSM, and Usual suspect approaches. Their method assigns the relevant files for over 70 % of the bug reports within the top 10 recommendations in Eclipse and Tomcat projects.

Rafi et al. [12] proposed an automated approach for finding and ranking potential relevant classes for bug reports. Their approach used a multi-objective optimization algorithm to find balance between minimizing the number of recommended classes and maximizing the correctness of the proposed solution. Based on the use of the history of changes and bug-fixing, and the lexical similarity between the bug report description and the API documentation estimated the correctness of the recommended classes. They evaluated their system on six large open-source Java projects. The experimental results showed that the search-based approach was better than mono-objective formulations (LS and HS). Their search-based approach can find the true buggy methods for over 87% of the bug reports within the top 10 recommendations.

Lam et al., [13] presented an integrating approach between deep neural network (DNN) and rVSM, and an information retrieval (IR) technique to locating and ranking potential relevant classes for bug reports. rVSM gathers the textual similarity feature between bug reports and source files. DNN is used to learn to relate the terms in bug reports to potentially different code tokens and terms in source files. The Evaluation of their approach was on real-world bug reports in open-source projects. Combining DNN with their new model achieved high accuracy of bug localization than the state-of-art IR and machine learning techniques.

The approach that was used to benefit from the "crowd knowledge" available in stack overflow to aid developers in their activities was presented in [14]. This strategy recommended a ranked list of question-answer pairs from stack overflow based on a query. The ranking criteria was based on the textual similarity of the pairs with respect to the query, the quality of the pairs, and a filtering mechanism that considers only “how-to” posts. They conducted an experiment about programming problems on three different topics (Swing, Boost and LINQ) frequently used by the software development community. The results showed that for Lucene+Score+How-to approach, 77.14% of the assessed activities have at least one recommended pair proved to be useful to solve a programming problem.

Fabio et al. [15] discussed developers abandoned from legacy developer forums to stack overflow platform, a lot of crowd-sourced knowledge is at risk of being left behind. They aimed to add to the body of evidence of existing research on best-answer predication. They did an experiment by using data from Stack Overflow to train a binary classifier. After that, they tested a classifier on a dataset retrieved from the legacy Doc using support forum. The findings showed that their model could find best answers with a good accuracy when all features are enabled e.g. answer up votes, number of sentences and answer length. Results gave a positive proof towards the automatic migration of crowd-sourced knowledge from legacy forums to modern Q&A sites.

Jacob Perricone [16] utilizes the network structure of Stack Overflow to recommend a set of related questions for a given input question. In particular, the project employs a modified guided Personalized Page Rank algorithm to generate candidate recommendations and compares the results to those recommended by Stack Overflow. Semantic similarity and tag-overlap were used to assess candidate recommendations. For a given recommendation set, the average text, title, and tag-overlap scores were calculated. These scores were then averaged across all trials of the experiment to yield a final score.

Daniel S. Weld et al., [17], investigate a new problem of systematically mining question-code pairs from Stack Overflow (in contrast to heuristically collecting them). They formulated the problem as predicting problem whether a code snippet is a standalone solution to a question. They proposed a novel Bi-View Hierarchical Neural Network that can capture both the programming content and the textual context of a code snippet (i.e., two views) to make a prediction. On two manually annotated datasets in Python and SQL domain, the framework substantially outperforms heuristic methods with at least 15% higher F1 and accuracy. Furthermore, they presented StaQC (Stack Overflow Question-Code pairs), the largest dataset to date of ~148K Python and ~120K SQL

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 September 2020 doi:10.20944/preprints202008.0265.v2

question-code pairs, automatically mined from SO using this framework.

Stefanie Beyer et al., [18] aim to automate such a classification of SO posts into seven question categories. As a first step, they have manually created a curated data set of 500 SO posts, classified into the seven categories. Using this data set, they applied machine-learning algorithms (Random Forest and Support Vector Machines) to build a classification model for SO questions. They then experimented with 82 different configurations regarding the preprocessing of the text and representation of the input data. The results of the best performing models show that their models can classify posts into the correct question category with an average precision and recall of 0.88 and 0.87 when using Random Forest and the phrases indicating a question category as input data for the training. The obtained model can be used to aid developers in browsing SO discussions or researchers in building recommenders based on SO.

Yun Zhang et al., [19] proposed a novel approach named RFEB, which recommends frequently encountered bugs (FEBugs) that may affect many other developers. RFEB analyzes Stack Overflow, which is the largest software engineering-specific Q&A communities. Among the plenty of questions posted in Stack Overflow, many of them provide the descriptions and solutions of different kinds of bugs. Unfortunately, the search engine that comes with Stack Overflow is not able to identify FEBugs well. To address the limitation of the search engine of Stack Overflow, they propose RFEB, which is an integrated and iterative approach that considers both relevance and popularity of Stack Overflow questions to identify FEBugs. To evaluate the performance of RFEB, they performed experiments on a dataset from Stack Overflow, which contains more than ten million posts. Finally, they compared this model with Stack Overflow's search engine on 10 domains, and the experiment results show that RFEB achieves the average $NDCG10$ score of 0.96, which improves Stack Overflow's search engine by 20%.

Table 1: Summary of related studies

Ref	Year	Method used	Data set	Results	Performances Evaluation
Xin et al.[11]	2016	- Ranking model - VSM	Benchmark datasets from open source projects: Eclipse -JDT - Birt - SWT	The learning rank approach achieved better results than the BugLocator , VSM, Ususl suspects on all six projects.	1) Eclipse : accuracy =80% ,MAP= 0.44 ,MAR=0.51 . 2)JDT: accuracy =80% ,MAP=0.39 ,MRR=0.51 . 3) Birt :accuracy=50% ,MAP=0.16 ,MRR=0.21.
Rafi et al., [12]	2016	- Multi-objective algorithms called NSGA-II - search-based algorithms	Benchmark datasets form open-source systems : - EclipseIU	NSGA-II is better than random search and the three mono objective formulations (lexical-based	1) EclipseIU : precision=82%,recall=79%, accuracy=80%. 2) Tomcat:

			-Tomcat -AspectJ -Birt -SWT - JDT.	similarity (LS) , history-based similarity (HS,.and (GA) on all the 6 systems.	precision=91%,recall=81% accuracy=90%. 3) AspectJ :precision=79% ,recall=86% ,accuracy =88%.
Lam et al., [13]	2017	Revised Vector Space Model (rVSM) + Deep Neural Network (DNN)	Benchmark datasets from open-source projects: Aspectj ,Birt ,Eclipe platform ,JDT,SWT,Tomcat .	DNNLOC achieves highest accuracy with the combination of relevancy via DNN, textual similarity via rVSM, and the metadata features.	1) TomCat :accuracy=80.4%, MRR=0.60,MAP=0.52 2) AspectJ : accuracy=85%,MRR=0.52,MAP=0.32. 3)
Eduardo et al., [14]	2016	Logistic regression classifier (LR)+ Normalized Discounted Cumulative Gain (NDCG)	Dataset from Stack Overflow (the version of March 2013)	The Lucene+ Score+ Howto approach achieved better performance than Google on Boost.	NDCGRelev=0.3583 NDCGReprod=0.5243
Fabio et al., [15]	2016	Alternating Decision Trees (ADT) classifier + information gain (IG)	1). Dataset from Docusign , it is a legacy forum. 2) Dataset from Stack Overflow	The model can find best answers with a good performance, when all features are enabled	accuracy ~90%, F=.86, AUC=.71.

4.1. Discussion

We noted from the previous state of the art that the most of studies like [11] [12] and [13] provide a ranking approach that leverages domain knowledge to locate a bug by ranking all the source files likely to contain the cause of the bug.

The researchers used the same benchmark datasets for evaluation in [11] [12] [13]. There is a similarity between [13] and [14], both of them used vector space model for ranking whereas [12] used Multi-objective algorithms called NSGA-II while [13] used Revised Vector Space Model (rVSM) and Deep Neural Network (DNN).

Data mining techniques that were used in these studies are different such as [15] and [18] that used classification, [16] that used a modified version of page-rank, [17] that used a novel Bi-View Hierarchical Neural Network algorithm and [19] that used a novel RFEB approach to recommends frequently encountered bugs. The researchers suggested in [13] and [14] to improve their approaches by leveraging additional types of domain knowledge and using the SVM ranking with nonlinear kernels. They evaluated their approaches in different datasets of programming languages codes. In this paper, we tried to take benefits from these suggestions in the design and the development of our proposed framework. Table 1 is showing a summary of all these related studies.

5. Proposed Approach

Our framework is using the concept of recommender systems to model the problem of mining stack overflow in order to find a solution for a bug report.

5.1. Overall Framework

By looking to the architecture of our model, we find that a bug report is issued as input and the ranked list of related best answers is recommended as output. First, when a new bug report is received, the preprocessing step is then started. The Information extracted from the bug report are then formulated as query and issued to the index of questions. A similarity measure between the query and a set of answers is calculated. The N best selected answers are then passed to the kernel of our model. The list of proposed solutions is then re-ranked by score. The learning to rank approach is applied to train a ranking model that use many features extracted from the Stack Overflow dataset. Figure 2 shows the overall architecture of our model framework.

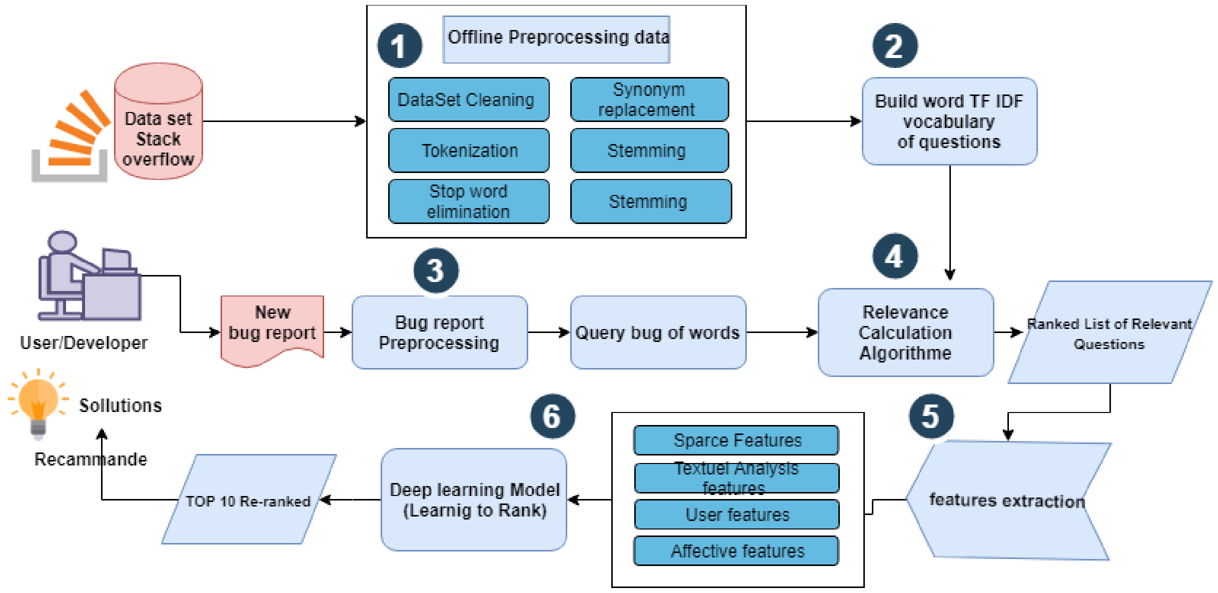

Figure 2: The overall architecture of our model framework

5.2. Data Preprocessing

Data Preprocessing is the most important task in data mining process particularly in text mining field where we are dealing with a huge amount of raw textual data. This phase has a great impact on the performance of the model. In our framework, text mining and NLP techniques are applied on two kinds of data:

- *SO data*: containing a large amount of textual data presented as posts (questions and answers) and Meta data about users. The goals from this preprocessing stage is to prepare this posts' data for the task of index building. This index will be used to train the ranking model.

- *Bug report*: contains a lot of information about the software bug processed by any developer or programmer. This data represent in our system what we call context (or query).

5.2.1 *Dataset Cleaning*

Before performing any type of preprocessing on the dataset, it is necessary to clean the data.We filter our data by deleting the unwanted parts.

5.2.2 *Tokenization*

This process split the sequence of strings into words. It removes all the punctuations marks from the input text data and returns words as tokens.

5.2.3 *Stop word elimination*

One of the fundamental ideas in the context of information retrieval is the removing of common words that appear frequently in the documents, but do not provide helpful information for the users' needs. These words called stop words can effectively decrease the retrieval rate.

5.2.4 *Stemming*

Refers to the procedure of transforming deferent variations of the same word to their stem, usually through stripping suffixes and prefixes such as (ed, ize, s, ing in English Language). Although stemming is a very commonly used process in information retrieval, it might cause false matching of some words with deferent stems to the same root.

5.3. Building TF-IDF Index of questions

The vector-space representation is a framework for representing raw or unstructured documents as vectors of terms. Using this idea, Vectors of term weights can be represented as one-row matrix representing the textual features of each bug report or question in Stack overflow dataset. The goal from building this index is to calculate the similarity between the bug report as a query and a set of questions from the OS dataset. The index of documents is searched to retrieve and rank the N first questions with high similarity.
The vast majority of work using vector-space representation makes use of the TFIDF representation of documents. The TFIDF representation is a heuristic metric that is used as a weight to represent each term feature of a given document.

$$TFIDF(t, D)=TF(t).IDF(t, D) \qquad (1)$$

TF(t) refers to the term *t* frequency in the current document *d*.

$$IDF(t, D)=log(\frac{D}{\mathrm{DF(t,D)}}) \qquad (2)$$

DF(t, D) refers the number of document where the term *t* appear and *IDF(t, D)* refers to the inverse document frequency, or the frequency of a term over the whole set of documents.

5.4. Similarity Calculation

Once a Bug report is submitted, the system will search for the *N* most similar documents among the knowledge bases. In order for our system to rank the results, it needs to rely on a metric to compare how similar a pair of documents (a bug report and a question) are.
For scoring the similarity between bug report and question, we used *Cosine* similarity, which is a standard form of measuring document similarity in vector space model. This measure is the cosine of the angle between two vectors and can be in the range of 0 (orthogonal vectors) to 1 (identical vectors). Cosine similarity between two vectors is calculated by the dot product of those two vectors and divide it by their magnitude.

$$Cosine\ (Q, D) = Q{*}D / |Q| * |D| \quad (3)$$

Based on cosine similarity between the bug report and each question, the top *N* questions are ranked. The system collect the answers that are directly related to the question from the SO database and return the whole set of answers which will be used again as input to the ranking model. The learn to rank process generate a new ordered set of answer after re-ranking them by score of pertinence to the original bug report.

5.5. Features Extraction

To select the best N relevant answers, our learning Model ranks answers based on a heterogeneous set of dense and sparse features. In our model, we focus on textual/embedding features of question-answer pairs and dense features based on three kinds of aspects: textual, community and affective.

5.6. Learning to rank model

Learning to rank in the context of Information Retrieval (IR) is a task used to automatically construct a ranking model based on the training data. This model is generally used to sort new objects according to their degrees of relevance, preference, or importance [20]. The ranking problem is defined as a derivation of ordering over a list of examples that maximizes the utility of the entire list [21]. We can consider this approach as very similar to classification and regression problem but ranking problems are fundamentally different. While the goal of classification or regression is to predict a label or a value for each individual document, the goal of ranking is to optimally sort the entire example list in a way that the examples with highest relevance are presented first.

In this way, we can naturally apply learning to rank to build ranking models to recommend solution for bug report based on Stack Overflow Data. We have two stage in learning to rank algorithm, the learning stage and the deployment (or testing) stage. The main task in this recommendation system is to train a ranking model $f(Br_i, A_j)$ where *Br* represents the bug report and *A* represented the associated Answers from stack overflow community.

recently, deep learning approaches were achieving better results compared to previous machine learning algorithms on tasks like image classification, natural language processing, face recognition and text mining field. Deep learning techniques have high power and capacity to resolve complex and non-linear problem . Neural networks can effectively incorporate sparse features like query or document text. Based on these advantages, we opted for the use of deep learning model to solve our ranking problem by training it from the textual data of stack overflow dataset.

6. Conclusion

This work aims to develop and propose a recommender system based on learning to rank approach and deep learning techniques. Our main idea was to investigate the use of Data mining on Stack Overflow to automatically suggest relevant solutions that fix software bugs and programming errors. This system will decrease the time generally spent by developers during the manual efforts for fixing bugs or during the consultation of Q&A websites.

We started our research by discovering this new area of data mining: mining software repositories. This new field will have an important impact in the future of software development. We also provided an overview of Text mining, NLP, Recommender Systems and Deep learning techniques.

In the part of contribution, we proposed an architecture of recommender system that contain baseline stage based on TF-IDF index and a learning to rank model based on deep learning to recommend relevant solutions for programing error and bug report.

For future work, we will try to test the approach of learning to rank using pair-wise or list-wise techniques. We will also try to improve our model performance using features that are more specific in the training phase. Other deep learning algorithms and techniques like CNN and LSTM will be also considered for future testing and comparison purposes.

7. REFERENCES

[1] A. E. Hassan, "The road ahead for Mining Software Repositories," in Frontiers of Software Maintenance, Oct 2008, pp. 48–57.

[2] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen. A topic-based approach for narrowing the search space of buggy files from a bug report. In Proceedings of the 26th International Conference on Automated Software Engineering, pages 263–272, 2011.

[3] Crowdsourcing. Wikipédia [Online]. 2020. [Accessed July,17 2020]. Available on: https://en.wikipedia.org/wiki/Crowdsourcing.

[4] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider. Mining Duplicate Questions in Stack Overflow. In Proc. of MSR 2016.

[5] Mining Software Repositories. Wikipédia [Online]. 2020. [Accessed July,17 2020]. Available on: https://en.wikipedia.org/wiki/Mining_software_repositories

[6] Data Mining Curriculum: A Proposal. KDD [Online]. 2020. [Accessed July,17 2020]. Available on: https://www.kdd.org/curriculum/index.html

[7] M. I. Jordan, T. M. Mitchell, Machine learning: Trends, perspectives, and prospects. Science 349, 255–260 (2015).

[8] Data science and machine learning. IBM Analytics [Online]. 2020. [Accessed July,17 2020]. Available on: https://www.ibm.com/analytics/machine-learning.

[9] Jeff Heaton. "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning - The MIT Press, 2016, 800 pp, ISBN: 0262035618". In: Genetic Programming and Evolvable Machines 19.1-2 (2018).

[10] Manning, C. and Schütze, H. 1999. Foundations of Statistical Natural Language Processing, MIT Press. May 1999.

[11] Ye, Xin & Shen, Hui & Ma, Xiao & Bunescu, Razvan & Liu, Chang. (2016). From Word Embeddings To Document Similarities for Improved Information Retrieval in Software Engineering. The 38th International Conference on Software Engineering, ICSE '16, May 14-22, 2016, Austin, TX, USA.

[12] Almhana, Rafi & Mkaouer, Mohamed Wiem & Kessentini, Marouane & Ouni, Ali. (2016). Recommending relevant classes for bug reports using multi-objective search. ASE 2016: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, August 2016 Pages 286–295, Singapore.

[13] N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, “Bug localization with combination of deep learning and information retrieval,” in Program Comprehension (ICPC), 2017 IEEE/ACM 25th International Conference on. IEEE, 2017, pp. 218–229.

[14] Campos, Eduardo & de Souza, Lucas & Maia, Marcelo. (2016). Searching Crowd Knowledge to Recommend Solutions for API Usage Tasks. Journal of Software: Evolution and Process. 28. 1-32.

[15] Calefato, Fabio & Lanubile, Filippo & Novielli, Nicole. (2016). Moving to Stack Overflow: Best-Answer Prediction in Legacy Developer Forums, 1-10, ESEM '16, September 08-09, 2016, Ciudad Real, Spain.

[16] Jacob Perricone, Question Recommendation On the Stack Overflow Network, Stanford University, 2017.

[17] Ziyu Yao, Daniel S Weld, Wei-Peng Chen, and Huan Sun. 2018. StaQC: A Systematically Mined Question-Code Dataset from Stack Overflow. arXiv preprint arXiv:1803.09371 (2018).

[18] Stefanie Beyer, Christian Macho, Martin Pinzger, and Massimiliano Di Penta. 2018. Automatically classifying posts into question categories on stack overflow. In the 26th Conference. ACM, Gothenburg, Sweden, 211–221.

[19] Y. Zhang, D. Lo, X. Xia, J. Jiang, and J. Sun. Recommending frequently encountered bugs. In International Conference on Program Comprehension, Gothenburg, Sweden, 2018.

[20] T. Y. Liu. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3):225–331, 2009.

[21] H. Li, Learning to Rank for Information Retrieval and Natural Language Processing, San Mateo, CA, USA:Morgan & Claypool Publishers., 2011.