Baltic J. Modern Computing, Vol. 6 (2018), No. 4, 349-362
https://doi.org/10.22364/bjmc.2018.6.4.03

# Aspect Based Construction of Software-Specific Words Similarity Database

Asif NAWAZ[1], Sohail ASGHAR[2], Muhammad Rizwan Rashid RANA[3]

[1]Department of Computer Science & Software Engineering, International Islamic University, Islamabad, Pakistan
[2]Department of Computer Science, COMSAT University, Islamabad, Pakistan
[3]University Institute of Information Technology, PMAS- Arid Agriculture University, Rawalpindi, Pakistan

asif.nawaz@uaar.edu.pk, sohail.asg@gmail.com, rizwanrana315@gmail.com

**Abstract.** There exist distinctive words that are used to express same semantics and as a result of this it has become hard to quantify the exact matching of words. To deal with this issue, past investigations endeavored to ascertain a likeness between distinctive pair of words. Conventional methodologies for computing word similarity are based on repositories like WordNet. It is a manually created lexical database and it processes semantic connection between various words. However, WordNet is a universally useful asset but wide range of words are not present in it and furthermore there exist an issue of identifying the meaning of words. Implication of words are diverse in WordNet when we utilize it in a textual framework. There exists a need of the refined approach that can gauge words resemblance in light of their co-occurrence. In this examination, we proposed an approach that registers likeness in text particular words, with the assistance of literary substance of various posts on StackOverflow. Our proposed strategy figures out word similarities in text by ascertaining the weighted co-occurrence in view of Computing Term Co-occurrence (CTC) and SentiWordNet. The exploratory outcome demonstrates that our system proposed an arrangement of words that are identified with text data is exceptional. Moreover, when it was compared with WordNet-based strategy named as WordNetres, it results with better outcomes.

**Keywords**. Social Media; Natural Language Processing (NLP); WordNet.

## 1. Introduction

With the quick advancement of computers in all fields of life, the volume of information and data increases with the advancement of data innovations. These innovations increment the volume of information by Microblogging locales, Blogs, E-Commerce sites and so on. It is assessed that consistently 2.5 trillion bytes data is delivered and 90% data of worlds is created in most recent two years. Increment in fast volume of data likewise named as 'Big Data' has made major issues i.e. how to locate the required data from trillions of data. To deal with this inquiry another term named as 'Big Data Retrieval' is conceived.

After the enhancement of software systems, stakeholders and developers typically make natural language artefacts (NLA) to communicate with one and another. Later on, the developers need to break down these NLA's to perform distinctive text building exercises (Haiyan, 2007). There are numerous examinations and these investigations proposed texted intends to build up these obligations. For instance, code search which takes a query as an input and returns us distinctive parts of the code that are identified with a particular query (Linstead et al., 2009). Indistinguishable bug report finding perceive diverse report records that characterize those issues that are same.However, it is composed in divergent practices by individuals (Wang et al., 2012). The essential movement in these strategies to figure out the similarity between two records. There are distinctive words in NL reports and these words have the same importance. Accordingly, to compute the resemblance of various records, it isn't conceivable to coordinate same words; thusly we have to ascertain the semantic separation among various words.

For instance, words like student, pencil, and paper are more comparative than Jupiter, road, computer and mountain. Figuring the semantic separation between various kinds of words is extremely basic for people. For machines and calculations, it is exceptionally harder. The NLP has dealt with this issue for quite a while (Jiang and Conrath, 1998). To enhance machine learning undertaking ascertain words similarity has a required assignment, e.g., data gathering (Chen et al., 2005) and content gathering (Islam and Inkpen, 2006). There are few errands identified with software engineering that are incorporated in these sorts of assignments that are programming particular (Marcus and Marcus, 2008; Sridhara et al., 2008). We can enhance these challenges with the assistance ofword similaritydata. In spite of the fact that a considerable measure of research work has been uncovered on the utilization of same words to the change of indexed lists when we apply code search (Runeson et al., 2007). Along with these lines, we can state that registering the comparability of various words is huge to text particular research.

For estimating exactness between the diverse matching of words NLP made WordNet (Miller, 1995). It is a database which is utilized for general purpose and it has bunch ofadjectives, verbs, adverbs and nouns into reasoning synonym groups. For ascertaining the semantic distance between two distinct words we can likewise utilize WordNet. In view of broadly useful nature of WordNet, it may not contain an expansive number of various words that are of textual context. For instance, extraordinary words i.e. programming or logical and database specific words like "localhost","cmd", "src", "WSDL", and so forth can't accessible in WordNet in light of the fact that these words are programming particular and database specific.

Also, in WordNetsome words are programming particular yet their semantic importance spared in WordNet repository is change. For instance, a word "Eclipse" in WordNet database is connected with the moon yet in programming perspective; it is an IDE (integrated development environments). Later on, another paper recommended that general similarity count established on WordNet can't give great outcomes and can't propose us same words in software perspective (Wu et al., 2009). Accordingly, we will build up an uncommon word likeness lexical database for groups identified with software engineering.

A great deal of research has been led to build up a word similarity repository that is especially for the group which is identified with software engineering. Yang and Tan, 2013 finished up with connected words in programming premise code. Another exploration led by Howard et al.(2013)extricates related verb sets accumulated from post and procedure marks. However various words that are identified with programming are additionally not present in the code, but rather placed in the various connected content substances, posts of forms, the reports identified with bugs, distinctive conferred logs and so forth. Moreover, there are few words that are in code, especially extraordinary commenting used to recognize something or utilized as a part of various strategies that are identified with few activities. Another examination directed by Wang et al., 2012 that assembles semantically same labels in FreeCode. In any case, they can just ascertain the likeness of various labels and not with the numerous ones that are in FreeCode. In our work, we will build a more refined word similarity database which will be utilized for various programming designed obligations on a wide range of related ventures.

In the event in which the substance of two words are same then it might be viewed as comparative. For instance, "tcp" and "customer" often show up in few sections, sentences, or online journals that depictnetworking. Keeping in view the end goal to recognize such excess, there is a need to build up another approach on the basis of the idea of word co-occurrence to compute the resemblance of two distinct words. We aggregate each word which evolves in the co-occurrence of a vector with some notable labels identified with software, different words and diverse programming labels which can relate each pair based on their co-occurrence.

Our new similitude metric$WordSim^{CTC}$,we endeavour to outline a semantic database for text corpus that is domainspecific and superior to WordNet. We used StackOverflow dataset that is a prevalent inquiry noting site and take its posts as info which incorporates a substantial number of words identified with software context. We additionally control the technique for labelling on account of its regularity which is maintained by countless data sites including SourceForge, FreeCode and StackOverflow. These labels are utilized for marking the key highlights of client created substance which are frequent terms that are programming particular. We utilize diverse posts from StackOverflow as semantic words to ascertain the similarity of various words.

We think about our technique that depends on$WordSim^{CTC}$, with an old word similarity database ascertained by a WordNet based strategy. Pedersen et al. (2004) named that as WordNet resource used for various words. We utilize SentiWordNet alongside with $WordSim^{CTC}$to get the upper 10 related words. In this we utilize ten people to judge the effectiveness of each and every technique by labelling the yield words with various scores to some degree. We assembled a wide range of words identified with programming setting that are not accessible in WordNet asset. Few words that are accessible in both $WordSim^{CTC}$and SentiWordNet DB,that is averagely computed and it is 51% higher than the average score of the WordNet asset. Our main contributions in this research are:

1. We build up a word similarity database that is programming particular utilizing 10,000 posts in StackOverflow.
2. We proposed another closeness construct strategy that depends on the technique for labeling and gathering a word based on co-occurrence. A similarity of words is ascertained by figuring the resemblance of their reliable characteristics and contexts.
3. We applied our proposed technique on various words that are programming particular with the assistance of ten humans. Our research demonstrates that strategy gives better outcomes when contrasted with WordNet. There are

55% words that don't show up in WordNet and other 45% percent that are accessible can't coordinate with its correct implications as indicated by the programming specific context.

Whatever remains of paper is sorted out as takes after. Segment 2 presents the Related Work. Segment 3 discusses about the Preliminaries. Segment 4 outlines the proposed framework depiction. A framework incorporates four noteworthy strides of proposed model including Dataset and Pre-processing. Segment 5 demonstrates the Experiments and results. Conclusion is tended to in area 6 and finally, there is a segment of References.

## 2. Related work

Acquiring similarity between two words is one of the straightforward NLP assignments. Numerous papers show the various techniques to degree of this similarity. A large portion of the mainstream existing methods contain a lexical database to ascertain similarities of words. Pedersen et al.(2004) have made a UI to allow clients to compute the semantic separation between words. They ascertain the likenesses of all sets of words in WordNet and freely (Porter, 1980).

Like these examinations, we additionally endeavour to figure out the similitudes in words. However, endeavouring to utilize, WordNet is a universally useful asset, we used to administrate the Normalised Google Distance(NGD), which is specific for the undertakings identified with programming build setting. Various strategies have additionally been recommended to naturally develop a dictionary (Chen et al., 2005; Falleri et al., 2010). They developed it on the distributional theory that embraces that wordin similar settings that is required to have the comparable sense. For introducing novelty in existing approaches wecentreon the software engineering group, preferably not the same as the general dataset we make for utilization of a dataset which is identified with software engineering.

Yang and Tan (2013) displayed the strategy for ascertaining the semantic closeness in programming source code document. They introduced a system that takes input code with a container of stopwords and produces the corresponded set of words. Exploratory outcomes demonstrate that this procedure is debugged in C and JAVA to judge the semantic related words with more accuracy. Later a comparable system is introduced by Howard et al., (2013) which compute the semantic scores from client remarks. They extricated 97 same verb sets from 150 strategies are tested arbitrarily from 36 Java codes over few spaces. In this examination, we likewise create semantically associated words. However, we analyse textual context that is programming particular opposed to breaking down code.Conventional techniques including machine learning and lexicon based strategies are especially utilized as a part of customary methodologies. Matveeva (2006) proposed the Vector Space Model (VSM) to figure the similarity between two vectors utilizing Cosine similarity..

### 2.1. Research Questions

There are some research questions:

1. How precisely our projected method is associated with the baseline method?

2. How universal our projected method is used to calculate the similarity of words?
3. How our proposed method is scalable?

As for as we see this from data recovery perspective, the primary inquiry figures exactness while the second computes review. A measure of soundness is called accuracy, while measure of culmination is called review. In last inquiry, there is a need to research an opportunity which creates WordSimSE DB from a product related words and the probability to increase WordSimDB SE by observing more surveys.

### 2.2. Preliminaries

We initially examine StackOverflow, which is extremely famous these days in term of questions and their answers. At this point we discuss about various prominent content pre-handling techniques, for example stop-word disposal, tokenization of data and stemming.

1. StackOverflow**:** It is one of the popular site on which we question about our problem. It offers a bridge for developers to support one another by answering and questioning. With more than 1.8 million people and over 5,000,000 queries on StackOverflow. Most of the subjects of StackOverflow are associated to software related tasks. In our research, we get dataset from StackOverflow to make a database that contains similarity between different words.
2. Word Co-occurrence: It is the idea of co-occurrence of words based on “context”, which talks about the nearby words of a specific word (Höst et al., 2000). For scope of word we used a sliding window that limits some context. The targeted words should be located in centre of the window. For example, a window having mass 7 would also contain the targeted word itself i.e. the three words having three words to its left and three to right. If word is located in the start, a size 5 sliding window only contains the target word and 2 other words that are appearing on right side of it.
3. The context of phrases and words varies as per their use in daily life of comparative semantics to some other phrases and words. In term of computer, "society" can be considered as "database" and "use" can be considered as a way that is used for database. For a particular query we usedGooglesearch engine. This concept is then further applied to construct a technique that automatically extracts all those pages that pertained a particular word association using Google page count. This technique is likely applicable in clustering, classification and language translation.

## 3. Proposed Model for Word Similarity Database

In order to get more successful results, we should make sure that our proposed approach performance should be equal or better than accepted solutions to software specific words similarity database construction. In the domain of software engineering especially, word similarity the proposed approach is compared with existing state-of-the-art approaches that have much better acceptance and credibility.

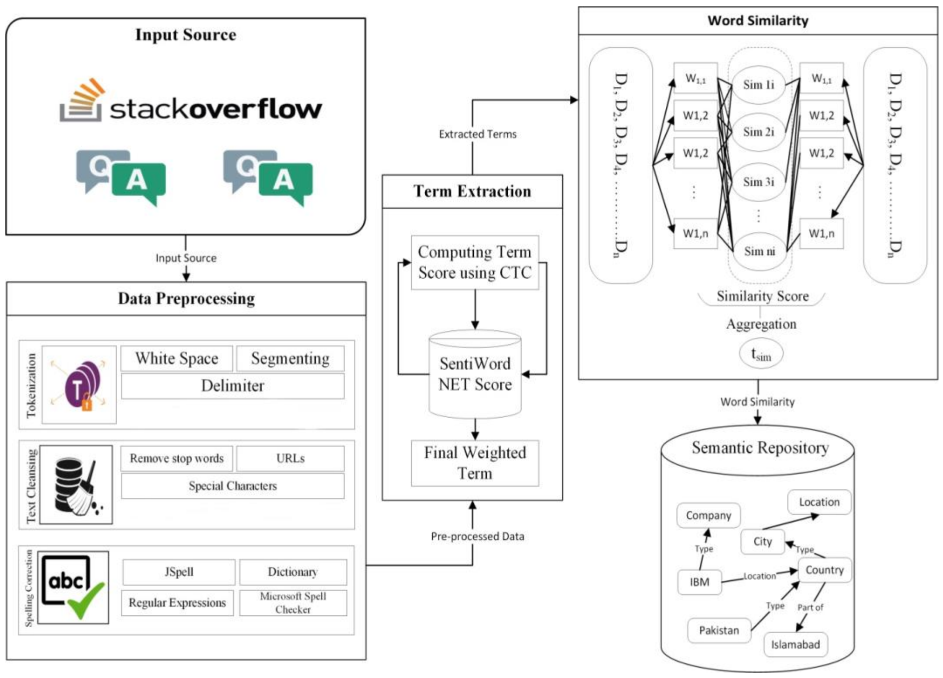

**Fig. 1.** Proposed model for Word Similarity Database

The proposed approach for $WordSim^{CTC}$ attempts to identify most appropriate words based on computational process. It extractssimilar words from reviews, blogs and users questions/answers repositories and thenassociates it with the word similarity database as shown in Fig.1. Different steps of proposed model will discuss in below section.

## 3.1. Pre-processing

Pre-processing can be considered as a key step in dataset pruning. In this module documents from social web possibly stack overflow and Facebook forums are taken as input. Such raw documents may contain text, code, tags and may also contain redundant or irrelevant data. Some redundant or irrelevant text snippetis discarded on the basis of following rules:

1. Universal Resource Locator (URL) will be removed because URL does not consider as a part of the job forgetting viewpoints.
2. We will remove every single word that does not start with English alphabet or a digit.
3. Common words like full stop, commas and punctuations etc. will also be eliminated by using a standard porter stemmer algorithm.
4. We will also remove those words that start with the symbol “@” because this symbol is used at the beginning of usernames and we are not taking users and their relationship in it. We will also remove words start with “#” symbol.

## 3.2. Computing Term Co-occurrence (CTC)

In this phase, each pre-processed document can be considered as a bag of words. CTC (Computing Term Co-occurrence) used to compute the semantic similarity between occur words with target words used in a document.Computation of Term Co-occurrence (CTC) is defined in equation (1).

$$CTC(t_1,t_2) = \frac{\text{high}\{logf(t_1),logf(t_2)\}-\log f(t_1,t_2)}{\log N-\text{low}\{logf(t_1),logf(t_2)\}} \qquad (1)$$

Where*f* $(t_1)$ is the number of pages containing occurrence of term$t_1$ and *f* $(t_1,t_2)$ containing association of both reported by Google. For the number of pages returned by Google we have to choose $N$ and it is apparent that by reducing the $N$, the CTC will increase. In this experiment some main properties of CTC that were applied are as follows:

1. The approximate value of the CTC lies between 0 and ∞, may be sometimes little bit negative if the Google search count irrelevant score or when it contains too much junk information for:
   *a.* If $t_1 = t_2$ or if $t_1 \neq t_2$ but the frequency $f(t_1) = f(t_2) = f(t_1,t_2) > 0$, *then*$CTC(t_1,t_2) = 0$.
   b. If the occurrence $f(t_1) = 0$ then for each term $t_2$ we have $CTC(t_1,t_2)$ and the$CTC(t_1,t_2) = \infty/\infty$.
2. The weight of CTC is almost nonnegative and $CTC(t_1,t_2) = 0$ for every$t_1$. For every pair $(t_1,t_2)$ we have$CTC(t_1,t_2) = CTC(t_2,t_1)$, e.g. x indicate the set of web pages holding one or more occurrences of$t_1$,e.g. choose $t_1 \neq t_2$ with x = y, formerly $f(t_1) = f(t_2) = f(t_1,t_2)$and$CTC(t_1,t_2) = 0$.

This association measure can be utilized to identify the most accurate co-occurrence of a particular term(see section 2.2 part 3). The main advantage of this approach is that doesn't require any background knowledge or any particular analysis of problem domain. Instead it automatically analyses all features through Google search using World Wide Web. In this phase term matrix is created that arrange each term along with corresponding target term. Term with minimum scorethan standard thresh hold i.e. 0.5 are discarded.

**Algorithm 1**. ComputeWord Similarity

```
Input  : All Pre-processed words ($W^{all}$)
         Output: $Words\ specific\ datbase\ WSD^{CTC}$
Initialize an empty $WSD^{CTC}$
for $w_i$ to $W^{all}$
   for $w_j$ to $W^{all}$
 $WSD^{CTC} = WSD(w_i, w_j)$
  end for
end for
```

### 3.3. Computation Terms Similarity

Generally, context words that are appearing with each term in a document are taken as a potential candidate for computing terms similarity in a document. To identify the context from a text, a sentiment dictionary or lexicon will be generated for each specific domain (SentiWordNet). Unit of resources are adjectives and verbs. Such as <No> + <adjective + verb>. Similarity calculation technique is applied to all sentiment units. We assign a unique name to this similarity score as $PC_{OCCR}$ (Point wise Occurrence).

$$P_{occur}(w_i , w_j) = log_1 \frac{P_1(w_i , w_j)}{P(w_i ), P(w_j)} (2)$$

Where

$w_i$=First word, $w_j$=second word

The optimized score is :

$$-\infty \leq P_{occur}(w_i , w_j) \leq \min[-logP_1(w_i) - \log p(w_j)(3)$$

Highest similarity score will be taken as final term score from text using equation (3). Generally $PC_{OCCR}$ scores will be calculated from whole document.

**Algorithm 2.** Compute Optimized Word Similarity

```
Input  :  All Pre-processed words ($W^{all}$)
          Output: $WordsspecificdatbaseWSD^{CTC}$
Initialize an empty $WSD^{CTC}$
Cache  = Empty bin for words.
for $w_i$ to $W^{all}$
   if($w_i$ not in cache)
       Compute scores from SentiWordNet of $w_i$
       Add score to Cache
Endif
    for $w_j$ to $W^{all}$
       if($w_j$ not in cache)
           Compute scores from SentiWordNet of $w_i$
           Add score to Cache
Endif
$WSD^{NGD} = WSD(w_i, w_j)$
 end for
```

## 4. Experiments

This section will take a closer look at the experimental results. Various experiments are performed on the different dataset to evaluate the performance of proposed approach. The detail description of each experiment is as follows

## 4.1. Datasets

We use three different datasets for experiments. Details of all three datasets are given below.

We develop a $WordSim^{CTC}$using the question and answer posts from StackOverflow. We get this data from MSR 2013 Mining Challenge (Demeyer et al., 2013). Our collected dataset is around 12 GB and holds all the posts that are produced from February 2014 to February 2018. We portray the data into different documents where every single document covers a question and all its answers. The description, title and all the tags of the question with their corresponding answers are mined and we save all this data in the document. We have collected 83,468 documents. Randomly we select sample 10,000 documents from all the dataset and use them to construct WordSimSE DB. All the tests are executed on an Intel Xeon X5460 3.26GHz server with 32.0GB RAM running Windows Server 2008 (32 bit). For the third step of our building procedure,we use 460 pairs of words gained from old work to adjust the weight parameters. We discover the better weights for α, β, and γ are 2.9, 2.1, and 1.5 respectively. It shows that other software tags are less important than popular software tags and other words are less important than the software tags. As a baseline, we use WordNet word pair similarity dataset. In this dataset we have billions of word pairs and size of this dataset is approximately 100GB. We compute the similarity of a word on the basis of Resnik matric (Roldan-Vega et al., 2013).

The popularity of micro-blogging is increasing day by day. People share their ideas on social media sites. Forums play an important part in social media sites, as forums allow users to share their ideas on any technique, method, issue etc. Software related forums are also present in vast number that is mainly used by developers or programmers. Usually, new ideasabout bug fixes and software fixes are discussed in these forums. We collected ten million comments from web related social media forums. These comments are extracted using Graph API (Weaver and Tarjan, 2013) from last 3 months.

Software repositories include packages related to software. Software companies and organizations maintain these repositories on their server. These repositories contain all information about software runtime errors, bugs, fixes and version details. These repositories are very much useful for checking any software reliability. We are using one of the biggest repositories named as ‘tera-PROMISE’ repository (Anwer et al., 2017). This repository deals with software engineering data. It contains millions of records about software engineering domain.

## 4.2. Results

Evaluation criteria are very much important in evaluating the results of any technique. We describe the results in two different and new criterias. These criteria are discounted cumulative gain (DCG)and Likert score (LS) (Jarvelin and Kekalainen, 2002). WordNet is one of the largest lexicons and also the base of many new generated lexicons. Usually, 45% words are present in WordNet lexicons which mean 450 words out of 1000 are present in WordNet. It also means WordNet returns only 450 software related words out of 1000. Our proposed approach returns1000 software related words out of 1000 which makes out approach far better than WordNet.

As mentioned earlier, WordNet provides the accuracy of 45% in terms of software related words. WordNet produces the average Likert score of 1.53 and our proposed approach has an average Likert score of 2.42. Our proposed approach gets the improvement of 60.18% which clears that our approach captures the software related words accurately by identifying the semantic meanings of words. Average Likert scores in tabular form are showing in Table 1.

**Table 1.** Average Likert Scores

| **Approach** | **Average Likert score** |
|---|---|
| WordNet | 1.53 |
| Proposed approach | 2.42 |
| Improvement | 60.18% |

Extraction of words with the ranking is one of the hardest tasks. Average Discounted Cumulative Gain is used to calculate the ranking of the extracted software related words (Wang et al., 2013), which implies that the most relevant document should be ranked first. Ranking of words always helps to use words in the right way by using their ranks. Table 2 shows the results coming from WordNet and from our proposed approach. It also shows improvement of 74.21% from Wordnet approach.

**Table 2.** Average Discounted Cumulative Gain scores

| **Approach** | **Average Discounted Cumulative Gain** |
|---|---|
| WordNet | 15.74 |
| Proposed approach | 28.62 |
| Improvement | 74.21% |

Table 3 shows the comparison of our proposed technique with some state-of-the-art techniques. These techniques are extracted from past studies named as WordNet and Castellanoset al. (2017). We extract 10,000 word pairs by analysing 3,000 reviews. As shown in Table 3 proposed technique extracts fewer word pairs from WordNet and it extracts almost double times pair of words from Castellanos technique.

**Table 3.** Comparisons in number of pairs

| **Approach** | **Pairs** |
|---|---|
| WordNet | 22,034,553,550 |
| Castellanos et al. (2017) | 5,636,534 |
| Proposed technique | 10,612,089 |

One of the research questions is scalability of proposed technique and the stability when reviews increase? To answers these questions we run proposed technique on 5,000, 10,000, 15,000, 20,000 and 25,000 reviews. Fig.2 shows the results when we run on different sets of datasets.

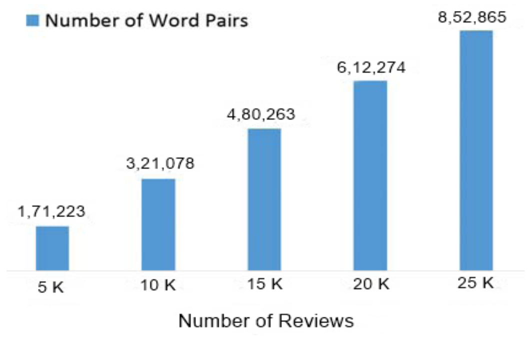

**Fig. 2.** Word Pairs

We also observed that how fast our proposed technique is working with some state-of-the-art technique. For this, we plot the runtime values with number of reviews. Data pre-processing, co-occurrence using normalized google distance and SentiWordNet are included in runtime. An experimental result shows that our proposed technique works 5 to 6 times faster than the other technique. Fig. 3 shows the comparison graph between proposed technique and state-of-the-art technique.

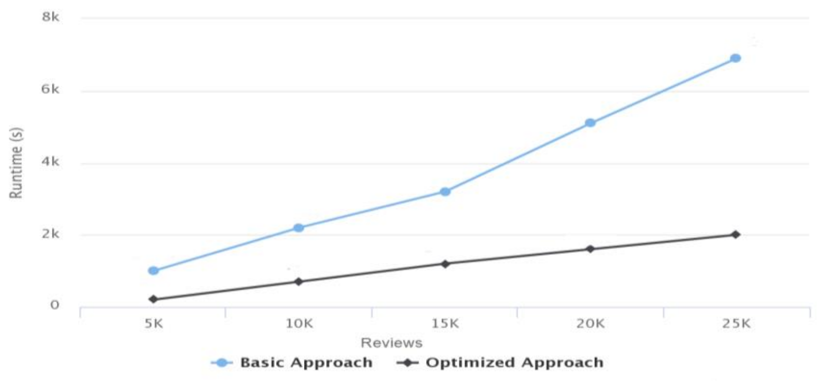

**Fig. 3.** Comparison between basic approach and optimized approach

Almost 30,000 comments are extracted from Facebookforums using Graph API. Comparison graph using basic approach and optimized approach is showing in Fig. 4. Reviews taken from social media forums are plotted on x-axis and time of a process is plotted on the y-axis. Results are extracted in term of the number of reviews like 5,000, 10,000, 15,000, 20,000 etc. Fig. 4 shows the graph that optimized technique also works well for the last amount of reviews and its results are improved in last reviews.

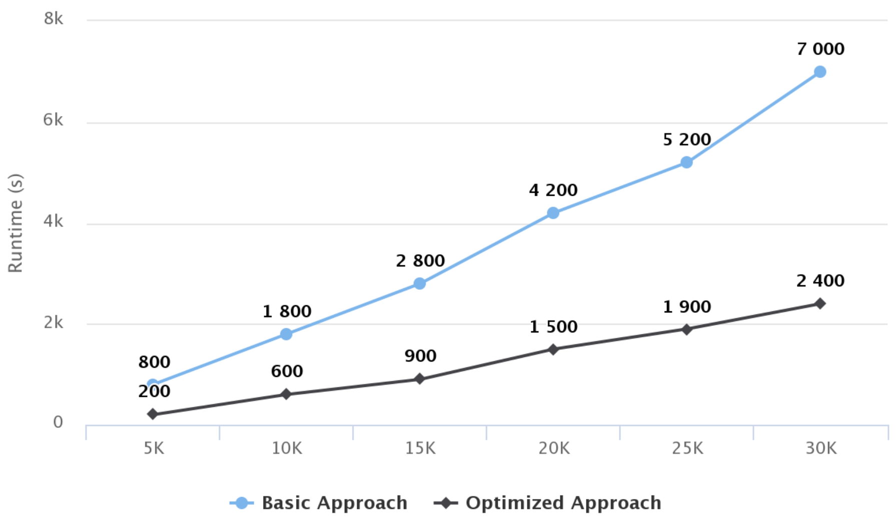

**Fig. 4.** Comparison of Facebook Reviews

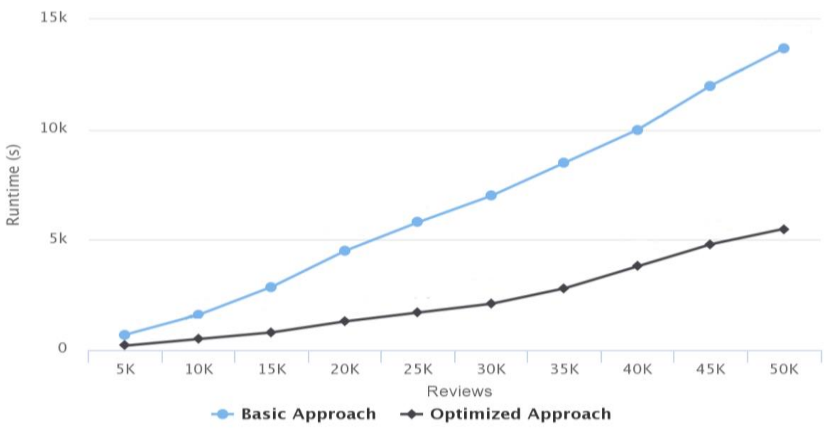

**Fig. 5**. Comparison of Software Repository

‘tera-PROMISE’ repository is one of the famous repositoriesthat iswidely used in many research papers. There are millions of records present in a repository but we pick fifty thousand sentences for our experiment. Experimental results are very promising, its runtime decreases as reviews or sentences increases. Experimental results are plotted in Fig. 5, where runtime is present in y-axis and reviews are present in an x-axis.

## 5. Conclusion

In this research, we proposed a method that automatically constructs software based term databases that save the common words of software engineering domain. We create a similarity metric named as WordSimSE based on question and answer posts in StackOverflow to calculate the similarity of different words based on their weights in co-occurrences with three different kind of anchors. We compare our technique with a WordNet-based approach named as WordNet res. From results, it seems that our technique produces better results than WordNet res in terms of average discounted cumulative gain (DCG) and average Likert score by more than 67% and 51% respectively. Our enhanced method can evaluate the similarity of more than 35 million pairs of words in less than 17 minutes by examining a 60,000 document dataset. In future, we are going to enhance it for larger$WordSim^{CTC}$ SE by executing it with more question and answer posts from StackOverflow. We also have a plan to allow open access to an expanded $WordSim^{CTC}$SE as a web service.

## References

Anwer, S., Adbellatif, A., Alshayeb, M., Anjum, M. S. (2017). Effect of coupling on software faults: An empirical study. *In Communication, Computing and Digital Systems (C-CODE), International Conference*, 211-215.

Castellanos, A., de Luca, E. W., Cigarán, J., García-Serrano, A. (2017). Partially squeezing the resources of the web of data towards applications. In *Intelligent Systems Conference (IntelliSys)*, 358-365.

Chen, L., Fankhauser, P., Thiel, U., Kamps, T. (2005). Statistical relationship determination in automatic thesaurus construction. *In Proceedings of the 14th ACM international conference on Information and knowledge management*, 267-268.

Chen, L., Fankhauser, P., Thiel, U., Kamps, T. (2005). Statistical relationship determination in automatic thesaurus construction. *In Proceedings of the 14th ACM international conference on Information and knowledge management*, 267-268.

Demeyer, S., Murgia, A., Wyckmans, K., Lamkanfi, A. (2013). Happy birthday! a trend analysis on past MSR papers. *In Mining Software Repositories (MSR), 2013 10th IEEE Working Conference*, 353-362.

Falleri, J. R., Huchard, M., Lafourcade, M., Nebut, C., Prince, V., Dao, M. (2010). Automatic extraction of a wordnet-like identifier network from software. *In Program Comprehension (ICPC), 2010 IEEE 18th International Conference*, 4-13.

Haiyan, C. (2015). Measuring Semantic Similarity Between Words Using Web Search Engines. *Computer Science*, vol. 42 issue 2, pp. 261-267.

Höst, M., Regnell, B., Wohlin, C. (2000). Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, vol. 5, issue 3, pp. 201-214.

Howard, M. J., Gupta, S., Pollock, L., Vijay-Shanker, K. (2013). Automatically mining software-based, semantically-similar words from comment-code mappings. *In Proceedings of the 10th Working Conference on Mining Software Repositories*, 377-386.

Howard, M. J., Gupta, S., Pollock, L., Vijay-Shanker, K. (2013). Automatically mining software-based, semantically-similar words from comment-code mappings. *In Proceedings of the 10th Working Conference on Mining Software Repositories*, 377-386.

Islam, A., Inkpen, D. (2006). Second order co-occurrence PMI for determining the semantic similarity of words. *In Proceedings of the International Conference on Language Resources and Evaluation*, 1033-1038.

Jiang, J. J., Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Järvelin, K., Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 422-446.

Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., Baldi, P. (2009). Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, vol. 18, issue 2, pp. 300-336.

Matveeva, I. (2006). Document representation and multilevel measures of document similarity. *In Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume: doctoral consortium* , 235-238.

Menzies, T., Marcus, A. (2008). Automated severity assessment of software defect reports. In Software Maintenance, 2008. *ICSM 2008. IEEE International Conference*, 346-355.

Miller, G. A. (1995). WordNet: a lexical database for English. Communications of the ACM, vol. 38, issue 11, pp. 39-41.

Pedersen, T., Patwardhan, S., Michelizzi, J. (2004). WordNet:: Similarity: measuring the relatedness of concepts. *In Demonstration papers at HLT-NAACL*, 38-41.

Porter, M. F. (1980). An algorithm for suffix stripping. Program, vol. 14, issue 3, pp. 130-137.

Roldan-Vega, M., Mallet, G., Hill, E., Fails, J. A. (2013). CONQUER: A tool for nl-based query refinement and contextualizing code search results. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference*, pp. 512-515.

Runeson, P., Alexandersson, M., Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. *In Proceedings of the 29th international conference on Software Engineering*, 499-510.

Sridhara, G., Hill, E., Pollock, L., Vijay-Shanker, K. (2008). Identifying word relations in software: A comparative study of semantic similarity tools. *In Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference*, 123-132.

Wang, S., Lo, D., Jiang, L. (2012). Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. *In Software Maintenance (ICSM), 2012 28th IEEE International Conference*, 604-607.

Wang, Y., Wang, L., Li, Y., He, D., Chen, W., Liu, T. Y. (2013). A theoretical analysis of NDCG ranking measures. In Proceedings of the 26th Annual Conference on Learning Theory.

Weaver, J., Tarjan, P. (2013). Facebook linked data via the graph API. *Semantic Web*, vol. 4, issue 3, pp. 245-250.

Wu, L., Yang, L., Yu, N., Hua, X. S. (2009). Learning to tag. *In Proceedings of the 18th international conference on World wide web*, 361-370.

Yang, J., Tan, L. (2014). SWordNet: Inferring semantically related words from software context. *Empirical Software Engineering*, vol. 19, issue 6, pp. 1856-1886.

Received June 21, 2018, revised October 7, 2018, accepted October 22, 2018