

# Quantifying Semantic Similarity of Software Projects Using Deep Learning

Akond Ashfaq Ur Rahman  
Department of Computer Science  
North Carolina State University  
Raleigh, North Carolina, USA  
{aarahman}@ncsu.edu

**Abstract**—Software re-usability can help software organizations to achieve rapid construction of software saving significant man hours. Finding similar projects within a software organization as well as within the open source domain, can help software teams to gather source code that can be re-usable helping them to construct software at a faster rate. The goal of this proposal is to help software teams in rapid construction of software using semantic similarity and deep learning techniques. The author uses available open source techniques to gather natural language tokens from software projects, and use them with deep learning techniques to quantify semantic similarity. The proposal uses Microsoft's Deep Semantic Similarity Model(DSSM) to quantify semantic similarity of nine open source software projects. The proposal also includes relevant empirical findings.

**Index Terms**—software repositories; semantic similarity;

## I. INTRODUCTION

The rest of the paper is organized as follows: we provide background information and related work in Section II. We present our methodology in Section III. We present empirical findings in Section IV. We discuss the findings our study in Section V. The limitations of our paper are presented in Section VI. Finally we conclude this paper in Section VII.

## II. BACKGROUND AND RELATED WORK

In this section we define the necessary concepts used in our paper and prior academic studies that are related to this paper.

### A. Definitions

### B. Semantic Similarity in Software Engineering

## III. METHODOLOGY

The author uses this section to describe the steps to perform relevant implementation and analysis. As shown in Figure ??, the study involved four major steps namely, collection of software repositories, extracting tokens, training models, and obtaining similarity scores. Finally the author ends this section by describing the experiments used in this study.

### A. Collecting Software Repositories

For analysis the author used popular Github projects written in three languages: C, C#, and Java. The author hypothesizes that semantic similarity might be different for different projects in different languages, and analysis of such aspect can bring better insight with respect to semantic similarity. Keeping this

assertion in mind, the authors has considered three different programming languages: C, C#, and Java.

The author selected the the first three projects that were 'trending' between March 01,2016 March 30, 2016, and had a size in between 1MB 100MB. The process repeated for three programming languages namely C, C#,and Java. Please note that 'trending' is a feature of Github that ranks Github repositories that gained popularity amongst Github users for a certain time period such as a day, a week, or a month.

### B. Token Extraction

The process of semantic similarity involves a collection of words or tokens using a which a semantic model can built on. To achieve this goal, the author used a two step process to gather necessary tokens from each of the software repositories of interest. These steps are presented below:

- The author used an open source tool called SrcML.NET to extract all the tokens from each repository. The SrcML.NET program takes the directory of each repository as input, and produces all tokens in one xml file.
- The author used another open source tool called Swum.NET that filters the tokens generated from SrcML.NET. The motivation of executing this step was to obtain natural language tokens from the tokens generated in the previous step by conducting the pre-processing steps:
  - convert camel-case and pascal-case tokens into natural language tokens
  - convert alpha-numeric tokens into natural language tokens such as converting to `_a_variable` to `a, variable`
  - remove tokens that are a length of one
  - remove tokens from the token collection that are stop words in the English language
  - remove tokens from the token collection that are language specific keywords such as void, int, and main.

After performing this step for each repository a .tsv file was created that was later used in training semantic models as well as evaluating the semantic similarity of software repositories.

### C. Training Semantic Models

The author has used the Deep Structured Semantic Model(DSSM) to quantify the semantic similarity between projects. According to Huang et al. [1] DSSM learns from a

query or a document which can be used to compute semantic similarity. DSSM trains itself by projecting semantically similar phrases that are close to each other, and projecting semantically dissimilar phrases that are further from each other.

DSSM also provides configuration options to create different training models from the same document. Amongst these configuration options the author has used two parameters namely *BATCHSIZE*, and *MAX\_ITER*. *BATCHSIZE* refers to how many training pairs can be used to train. *MAX\_ITER* refers to the total count of iterations DSSM will use to create the training model.

Similar to Yih et al. [2] the author performed the following actions to create the training model for each configuration:

- Shuffle the query pairs using the *WordHash* utility with the *shuffle* flag. Yih et al. [2] performed a similar step to .
- Following Gao et al. [3] the authors generate the sequence of letter trigram features using the *WordHash* utility with the *pair2seqfea* flag.
- The generated sequences of letter trigram features are converted to binary files using the *WordHash* utility with the *seqfea2bin* flag.
- Next, the noise distribution of the training data is calculated using the *ComputelogPD* utility.
- Using the a configuration file, and the *DSSM\_Train* utility, the author created training models that are used to perform the analysis of this paper. The author describes which model is used for what experiment in Section III-D.

#### D. Experiments

The author has designed four experiments to evaluate the similarity between two projects. In these experiments the author has varied the training model with respect to DSSM configuration parameters as well as the documents that are using. Table ?? presents the experiments and the parameters that were changed. The experiments are referred by their names in Section IV. In each of these experiments, similar to He et al. [1]'s approach the author has used Cosine similarity to quantify the vectors of tokens amongst two projects. In each of the experiments, the nine projects are compared in a pair-wise manner i.e. each of the nine projects are compared to itself, and the rest of the eight projects of interest.

### IV. RESULTS

[?]

A. Software Repositories

B. Token Extraction

C. Similarity

### V. DISCUSSION

In this section we discuss our findings by stating the implications of our findings for programmers and researchers.

• **Observation-1:** .

• **Observation-2:** .

• **Observation-3:** .

### VI. LIMITATIONS

We discuss the limitations of our study in this section as following:

### VII. CONCLUSION

#### REFERENCES

- [1] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 2333–2338. [Online]. Available: <http://doi.acm.org/10.1145/2505515.2505665>
- [2] W. Yih, M. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, 2015, pp. 1321–1331.
- [3] J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng, "Modeling interestingness with deep neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014, pp. 2–13.