Quantifying Project Similarity Leveraging Deep Semantic Similarity Model(DSSM)

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

***Abstract*—**
***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 relavant 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 tosemntic similarity. Keping 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 repsoitories that gained popularity amongst Github users for a certain time period such as a day, a week, or a month.

B. Token Extraction

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

- The author used an opne source tool called SrcML.NET to extract all the tokens from each repsoitory. 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 genrated from SrcML.NET. The motivation of excuting this step was to obtain natural language tokens from the tokens generated in the previous step by condcting the pre-processing steps:
  - convet 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 languae 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 Dep Structured Semantic Model(DSSM) to quantify the semnatic similarity between projects. According to Huang et al. [1] DSSM learns from a query or a document which can be used to compute semnatic siilarity. DSSM tarins itself by projecting semantically similar phrases taht are close to other, and projecting semantically dissimilar phrases that are further to another.

DSSM also provides configuration options to create different training models from the same document. Amongst these configuraion options the author has used two parameters namely *BATCHSIZE*, and *MAX_ITER*. *BATCHSIZE* refrs to how many trainign pairs can be used to train. *MAX_ITER* refers to the the

total count of iterations DSSM will use to create the tarinign model.

Similar to Yih et al. [**?**] the author performed the follwoing actions to create the training model for each configuration:

- Shuffle the query pairs using the *WordHash* utility with the *shuffle* flag. Yieh et al. [2] performed a similar step to .
- Following Gao et al. [3] the authors generate the sequence of letter trigarm features using the *WordHash* utility with the *pair2seqfea* flag.
- The generated sequences of letter trigarm features are converted to binry files using the *WordHash* utility with the *seqfea2bin* flag.
- Next, the noise distribution of th 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 tarining model with respect to DSSM configuration parameters as well as the documents that are using. Table **??** presents the experiemnts and the parametrs tht were changed. The experiemnts are referred by their anmes in Section IV. In each of these experiemnts, similar to He et al. []'s approach the author has used Cosine similarity to quantify the vectors of tokens amongst two projects. In each of the experiemnt, the nine projects are compared in a pair-wise manner i.e. each of the nine projects are comapred 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 follwowing:

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.