Lancaster University

School of Computing & Communications

Computer Science Bachelor thesis

Mining categorised software repository data for high accuracy code-to-comment translations

---

Author
Jacob Knight

Supervisor
Dr. Yehia Elkhatibe

February 20, 2020

1 Abstract

2 Introduction

Code documentation is a pivotal step to ensure the longevity and maintainability of software, with poor or no documentation it is not uncommon to see fragments of code become lost in translation, void of any intelligible meaning that the original developer intended. A well documented, consistent and complete piece of code can be a great tool in communicating software systems, making future iterations and updates fluid to roll out. The benefits of well documented code are indispensable for future developers and maintainers, so why isn't this practice conventional in software development? In general, there are no defined universal standards for producing documentation, combined with the price and time pressures that are prevalent in the software industry, documentation has become a secondary priority [1].

The proposed system aims to alleviate the pressure of producing code comments by implementing an NMT model [2] specifically a sequence to sequence variation [3], to automate the documentation process. Unfortunately NMTs are known to be computationally expensive and struggle to translate words that are rare within its known vocabulary [4]. To combat these issues two systems will be used. For the issue of computational expense; a web server will act as a proxy for translation inference, with several pre-trained models running on a dedicated system processing the translation. For the more immediate obstacle of unknown vocabulary, referred to ask "UNK tokens" in inference [5], a more elaborate approach will be taken.

With large datasets containing hundreds of thousands of unrefined code-comment pairs, a high frequency of UNK tokens is to be expected. A study conducted in 2016 [6] with similar scope to the proposed project, extracted code-comment pairs from the 6 highest rated python repositories, with each repository contrasting significantly in utility and topic. The dataset resulted in a relatively low BLEU score of 7.5, with the evaluation stating:

"The dataset is the result of combining 6 different software projects which each have a slightly different style in their comments. The wording consistency varies heavily as a result, and the amount of unique tokens is almost tenfold" - Tjalling Haije.

The main objective of this project as stated in the report title, is to analyse and define language pragmatics in hopes to increase model accuracy and avoid the situation encountered above, this specifically relates to the issue of unknown tokens and rare words during the vocabulary building process. The topic of dataset categorisation is discussed further at chapter 5.1 page 7.

3 Related work

Understanding previous works on automated documentation is critical in working towards a more attainable application of AI generated comments. Analysing a report not only gives further understand to the research problem at hand, but also highlights potential gaps within their methodology, allowing for one to identify their own research in the context of existing literature. This section will explore four different areas of automated documentation. Firstly a report of stark similarity, investigating how sequence-to-sequence fairs for the task of translation from a non-natural language to english. Secondly a review on a piece of literature with contrasting technologies and its relative success, finally an overview into the current issues surrounding Neural Machine Translation and how this report aims to approach them.

3.1 Automatic comment generation using sequence to sequence models

Literature and research surrounding the topic of automated documentation has seen a substantial rise over the past decade, a wider variety of machine learning techniques are being applied to create more accurate translations of source code. Some of the earlier work surrounding this topic took a similar approach to that of this report, utilising sequence-to-sequence models and mined Github data to evaluate the effectiveness of NMT translation.

The literature in mind by Tjalling Haije [6], explores the accuracy of sequence-to-sequence models across two datasets: a pseudo-set, containing annotations of pseudo level source code from the Django framework, and a lower-level corpora, containing code-comment pairs from 6 Github-mined repositories. The application of natural language techniques such as word embedding and bucketing work reasonably well for the pseudo dataset, returning a BLEU score of 41.6, however when applied to the secondary set containing source code from a selection of sources, the score dips to 10.7.

Comparing the two datasets suggests the summarization of source code and its contextual descriptiveness is impacted by the variance in functionality and comment style, and while evaluation scores seem high for the pseudo set, it appears that comments that are written for high level structured code offer no assistance in code comprehension and is therefore less practical for working situations. The pseudo set contains 18000 pairs, approximately 1.5x less than the collective Github set, the functionality and comment style of the source code is also refined to a singular framework, as opposed to six. Considering the difference in dataset size and evaluation score (a larger dataset size should result in higher accuracy scores), it's reasonable to assume comment accuracy is directly correlated with documentation style and repository functionality.

By contrast, this project aims to implement multiple categorised datasets

for training on the Sequence-to-Sequence model, in hope to achieve more complete and accurate translations. Whilst Haije makes a concrete attempt at exploring the feasibility of Neural Machine Translation for automated documentation, his results show conflicting conclusions. He presents that translation of logically structured code to a natural language is entirely possible assuming the dataset is of high quality, however the impracticality of seq2seq is emphasised when applied to a genuine dataset containing all the nuance and semantical variance found within a working repository.

3.2 Code cloning

More research into automated documentation, namely the popular paper/project CloCom by Edmung Wong et al. [7] explores the semantic importance of natural language on this issue. The project aims to address the complexities of analysing large sources of repository comment data, an alternative to their previous method in which StackOverflow Q&A data was mined. In total CloCom evaluates 28 projects containing over 3.8 million lines of code. The project takes the approach of matching inputted code to find structural similarities between software repositories using code clone detection techniques. The cloning techniques used within CloCom stems from their self-implemented token-based algorithm, which applies token by token cloning, achieving a time complexity of O(n * m) where n is the number of lines of code from the input project and m is the number of lines of code in the database. This makes use of an Abstract Syntax Tree parser for the tokenisation of source code which consequently restricts this technique to a singular language.

The system proceeds to leverage information from all returning code segments that match the input, disregarding any code comments that contain project specific information or fail to have semantic similarity. Once data has been accurately filtered the remaining code comment candidates are ranked and the most suitable project is selected. The translation process of CloCom is as follows: The application takes two inputs, one containing software repositories for comment extraction and another of target repositories of code that is to be automatically documented, returned as a list.

CloCom achieves average results, however compared with the previous sequence-to-sequence implementation a different evaluation metric is implemented. Explained briefly, CloCom ranks comments based on two heuristics, 1) the closeness of the translation regarding the context of the code and 2) the conciseness of the comment. The results of this research generated a total of 359 comments across 281 unique code segments, which averaged around 17 comments per project. The yield of these results is surprising low considering the 3.8 million lines of code used. The results state the following:

With 70.3% of comments being rejected by CloCom's cloning system,

Good	Requires modifcation	Bad	Total words
85	21	253	359

and with an entirely different evaluation metric, it's hard to conclude whether the system is a success comparatively to that of Haije's [6] sequence-to-sequence implementation. With optimal parameters of working set data, Haije secured a BLEU score of 10.7%, the heuristically evaluated successful comments by Edmung et al score 20.7%. Other variables are significant in determining the more applicable translation system between the two reports, such as project size and language, however any ultimate decision still seems inconclusive.

Despite CloCom utilising an Abstract Syntax Tree (AST) tokenization cloning system, the two reports do share similarities. Mainly, as stressed consistently throughout this report, is the lack of refined data. Both studies use corpora with no correlating functionality in their source code, with the only recurring property being either the training language or highly rated Github repositories. It would be interesting to see the correlation of categorial data for both AST and sequence-to-sequence translation models, unfortunately only sequence-to-sequence translation will be the focus of this report.

3.3 Six Challenges for Neural Machine Translation

4 Theoretical framework

To give more background into this report and to establish a solid understanding of the foundations of translation tasks, this section will explore the theoretical frameworks used within this project. Subjects that account for the bulk of translation tasks include, Natural Language Processing, Neural Machine Translation and Deep learning. Other subjects that tie in more specifically to the task at hand such as sequence-to-sequence models and their respective mechanisms will also be explained.

4.1 Natural Language Processing

A branch of AI that underlies the foundations of this project is Natural Language Processing, usually abbreviated to NLP. NLP is the study of interactions between computers and human natural languages, arguably the basis of this report, and therefore a topic that deserves to be touched on briefly. While NLP defines a broader category of study, its applications into translation tasks are undoubtedly important. An example of an NLP process that falls within the scope of this project is Semantic role labeling (SRL). SRL aims to give a semantic role to a syntactic component of a sentence [8], i.e. detecting semantic arguments relating to the predicate or verb of a sentence. For a comment delineating the functionality of a method that reads "A returns X", the goal would be to acknowledge the verb "returns" as the predicate, "A" as the agent, and "X" as the object or theme, these are called thematic relations and help express and identify roles within sentences. While this is a fairly condensed overview of the procedures of SRL, it's application in conjunction with Neural Machine Translation could lead to higher accuracy translations by producing semantic analysis on frequently used verbs such as returns, open and send. This is not used within this project however could be a consideration for future works of automated documentation.

Many NLP techniques will be used throughout this project and explained in depth. Lexical analysis or word segmentation such as tokenisation, byte-pair encoding and topic segmentation will be discussed in the methodology of this report.

4.2 Neural Machine Translation

4.3 Sequence-to-Sequence model

4.4 Abstract Syntax Trees

4.5 Data mining

5 Methodology

The following section will cover the step by step processes involved in achieving a translation from source code to natural language, mainly covering the acquisition of data, model construction, configurations and training techniques. The pitfalls and challenges faced during this phase will also be discussed, such as the difficulties when working with messy data, complex models and the restrictions of third party APIs.

5.1 Repository topics

To begin an analysis on the impact of categorised data on translation accuracy, one must first amass a corpora of data that adheres to a recurring pattern. Initially the proposed method was to separate datasets by language, however the disparity in functionality within each language meant comment semantics were too varied for a categorical dataset. Overcoming this problem required a monolingual approach in which comment context and style were clearly differentiable between datasets. The approach used to formulate said datasets utilised Github's search API [1] and labelling system; topics.

Topics allow for a network of similar repositories to be easily discovered, creating subject-based connections between one another [9]. In conjunction with the Github API, topics act as an extremely powerful tool to refine projects by type and technology, and in turn allows us to construct datasets that are uniquely distinct.

The initial dataset was built with python using both the Git and Github API allowing topics and their respective repositories to be cloned locally. To begin the extraction of topic information a repository object containing the metadata of 1500 python repositories was initatilised via the Github API. Each repository in the object-list was iterated through; if topics were present they were tallied and stored in a dictionary, Git URLs corresponding to tallied repositories were also noted. Upon completion, the 5 most popular topic-dictionary entries were exported to JSON format, the outcome of topic extraction is as followed:

Deep learning	Machine learning	Tensorflow	Pytorch	Linux
95	82	68	39	29

This data gives a rough estimate as to what type of python repositories are most prevalent on Github, allowing for a larger corpus of categorised

[1] https://developer.github.com/v3/repos/

repositories to be cloned. NMT models have steeper learning curves regarding training data, with low resource settings, a worse quality translation is to be expected comparatively to that of a larger dataset [10], hence the reasoning for prioritising topics with the highest number of correlating repositories.

To proceed with the construction of our initial datasets, the above-mentioned Git URLs and their relating topics were iterated through and cross-referenced against the top five topics, if the pair of topics intersected, that repository was cloned to a directory named after the topic. In the case where a repository held topics that matched 2 or more of the top 5, this was again noted and the repository was cloned into the directory with the smallest amount of data. Figure 1 shows how the top 3 topics intersect with one another, a small detail, however a consideration for any results showing similar or anomalous correlations.

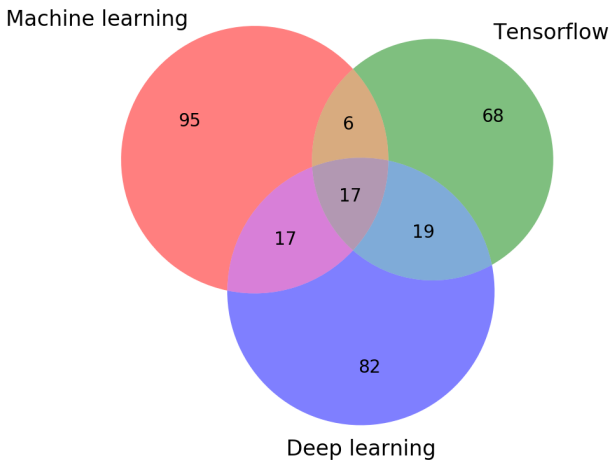

Figure 1: Venn diagram displaying intersected categories for repository cloning

5.2 Datasets

The core of this project revolves around the implementation of quality data to train our sequence to sequence model. To gain an overview into how this project utilises such data to obtain a comprehendible natural language translation, the acquisition, preprocessing and post-processing techniques will be covered, hopefully shedding light on how inputted data i.e. source code, reaches its corresponding output.

In this project two primary datasets are used: a comment set, containing code-comment pairs consisting of comments delineated by "#", and a docstring set, containing method documentation and method code pairs, both datasets are exclusively in python. Each primary set is further extrapolated to compose 3 additional sets which correspond with a categorised topic, the previous subsection Repository Topics outlines topic occurrence found from repositories in the comment-set. The final list of datasets are as followed:

Dataset	No of pairs
comment-set	
comment-set-deep-learning	
comment-set-machine-learning	
comment-set-tensorflow	
docstring-set	
docstring-set-	
docstring-set-	
docstring-set-	

Amongst the two sets, the docstring-set is significantly larger, it contains a collection of 500,000 pairs and is provided by GitHub as an open source resource in aiding research for code retrieval using natural language [11]. The comments from this data provide a more convoluted synopsis of method code in comparison to its code-comment counterpart, which describes code at a higher level with only singular lines of code and corresponding comments being extracted. With entire methods being translated to a natural language, the complexity of translation becomes a challenge to Neural Machine Translation systems. Research on sentence length for neural machine translation by Philipp Koehn and Rebecca Knowles (2017) [10] shows that quality of translation for sentences longer than 80 tokens significantly worsens. Method code extracted from this dataset can exceed 200 tokens, meaning pre-processing techniques to trim sentences of length greater than 80 tokens will be required. Alternatively for longer sentence pairs, Statistical Machine Translation (SMT) is regarded as a superior system and is worth mentioning as a consideration for future works.

References

[1] L. C. Briand, "Software documentation: how much is enough?," pp. 13–15, March 2003.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[3] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," *CoRR*, vol. abs/1703.01619, 2017.

[4] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[5] M. García-Martínez, L. Barrault, and F. Bougares, "Neural machine translation by generating multiple linguistic factors," *CoRR*, vol. abs/1712.01821, 2017.

[6] T. Haije, B. O. K. Intelligentie, E. Gavves, and H. Heuer, "Automatic comment generation using a neural translation model," *Inf. Softw. Technol.*, vol. 55, no. 3, pp. 258–268, 2016.

[7] E. Wong, Taiyue Liu, and L. Tan, "Clocom: Mining existing source code for automatic comment generation," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 380–389, March 2015.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Computing Research Repository - CORR*, vol. 12, 03 2011.

[9] S. Frendt, F. Jennings, J. Cool, and J. Epling, "Introducing topics," Jan 2019.

[10] P. Koehn and R. Knowles, "Six challenges for neural machine translation," *Proceedings of the First Workshop on Neural Machine Translation*, 2017.

[11] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt, "Codesearchnet challenge: Evaluating the state of semantic code search," 2019.