Extraction of UML Class Diagrams from Text with Only Machine Learning

Song Yang (`song.yang.1@umontreal.ca`)

Abstract

UML is a modeling used in software engineering. Its translation directly from English description is highly sought. This paper attempts to use a neural-only method to translate from English to UML class diagrams written in yUML, a textual programming language. Because the training data is a small toy data set, the model failed to learn a meaningful translation. The project's repo is available here.[1]

1 Introduction

Unified Modeling Language (UML) class diagrams are a concept introduced along with other UML diagrams in the 1990s. Despite being standardized in 2005, there is no agreed upon formal semantics. Because human language is semantically flexible, it is likely that existing NLP translation models can translate into UML.

2 Related Work

Non-neural approaches (Ibrahim and Ahmad, 2010) uses OpenNLP, a parser with deterministic rules for extracting information from text. This rule-based approach is also used by (Jyothilakshmi and Samuel, 2012).

Mixed approaches (Saini et al., 2020) uses a mixture of NLP parsing and machine learning to extract information from text. The machine learning is done on heavily annotated text in order to learn part-of-speech (POS) tagging. The entire pipeline is complex and users are required to interact with the system to get the UML they want.

3 Task and Data set

UML class diagram generation from text The task of this paper is to translate human language text into UML class diagrams. This process is a typical software engineering assignment in undergraduate studies. An automated procedure for UML class diagram generation can help with understanding specifications written in English.

Data set The data set for this paper comes from a small handcrafted set of UML diagrams and their English description.[2] Every class diagram is paired with an English prose, typically a small paragraph of text. The UML class diagrams are encoded using a textual representation called yUML.[3] The language of yUML functions like LaTex. That is, it is a purely visual language that renders into an image.

The handcrafted data set is split in a train-test-valid split of 39-13-13, which is 60-20-20. Each data sample is a typical construction of a UML class diagram feature. For example, there are 3-1-1 examples of how inheritance is described in English and how they are written in yUML syntax.

4 Methods

Pre-processing In order to train a neural machine translator, the source and target languages must be tokenized. In the case of the source English text used in the handcrafted data set, the tokenization of English is well-known for machine translation. Several tokenizers exist, namely Spacy[4] and NLTK[5]. We opt for NLTK's simplest word tokenizer.

However, for the paired UML diagram written in yUML, there is no such tokenizer. Because the lan-

[1] `https://github.com/XsongyangX/ift6755-project`

[2] `https://github.com/XsongyangX/UML-hand-class-diagrams.git`

[3] `https://github.com/jaime-olivares/yuml-diagram/wiki`

[4] `https://spacy.io/api/tokenizer`

[5] `https://www.nltk.org/api/nltk.tokenize.html`

guage of yUML resembles the looks of ASCII art, an English-language tokenizer cannot be used. As such, we build our own tokenizer from the language specifications of yUML, by using their publicly available context-free grammar. (Batory, 2017) The yUML tokenizer is built using Flex/Bison, standard tools for compiler construction.

Neural model The neural model used to perform the machine translation from English to yUML is attention-based. (Luong et al., 2015) Attention in neural machine translation is the idea of letting a model focus on different parts of the input. This attention changes depending on which output token the model is currently at. Attention gives the model a chance to look beyond the last RNN state, thus making the model consider the context of the entire input at once.

The code for the attention-based model is freely available on GitHub here[6] and here[7].

5 Evaluation

BLEU BLEU (bilingual evaluation study) is a popular metric for evaluating natural language translations. However, this project deals with unnatural language translations. This difference makes BLEU scores less reliable.

Manual Since UML is made for communication, human evaluation is still needed.

6 Results

After one to two hours of training the neural model, the training loss has decreased to 0.07. The experiment was run on Google Colab GPU.

BLEU The BLEU score is 20, which is on par with similar state-of-the-art model for human languages (Post, 2018). This metric is rather meaningless as we take a look at the translation output.

Manual For example, the following "*All lamps have a lightbulb . A lightbulb has a filament .*" is translated into

```
[ Wage ] ^ [ Salaried ]
[ Wage ] ^ [ Contractor ]
```

The translation does not match the topic discussed. Every other translation has the same problem. Sometimes, an empty translation is given. Thus, the neural model has failed.

[6]https://github.com/AotY/Pytorch-NMT
[7]https://github.com/pcyin/pytorch_basic_nmt

7 Analysis

The shortcomings of neural translation are due to the lack of more extensive data. Since the translations are just UML diagrams from other training data, the model has learned associations between data points. It has not learn how the data points themselves separates into smaller semantic pieces. This situation is the equivalent of replying a sentence from a tourist guide to a speaker of a foreign language.

One of the translations have matching topic, but the wrong UML construct. This happened on a data point whose English prose are describing very similar things with the same words. This rather acceptable translation hints that the model can improve given more data on similarly worded topics.

8 Conclusion

Neural translation from English to UML class diagrams is not working in this project. Despite achieving a good BLEU score, the translation is irrelevant. Possible explanations include the lack of substantial data to train on.

8.1 Future work

For future work, it may be interesting to look at pre-trained models such as BERT and XLNet, which already digested a large corpus of the English Wikipedia. This may ease the need on extra data.

Because UML is not a natural language, a directly coded rule-based parser may be needed. After all, UML is only an abstract grammatical representation of English. A grammatical parser trained on the Penn Tree Bank may be better than an assumption-free neural model.

References

Don Batory. 2017. *YUML and YPL Database Manual*. University of Texas, Austin, TX, USA.

Mohd Ibrahim and Rodina Ahmad. 2010. Class diagram extraction from textual requirements using natural language processing (nlp) techniques. In *2010 Second International Conference on Computer Research and Development*, pages 200–204. IEEE.

M. S. Jyothilakshmi and P. Samuel. 2012. Domain ontology based class diagram generation from functional requirements. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 380–385.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Rijul Saini, Gunter Mussbacher, Jin LC Guo, and Jörg Kienzle. 2020. Domobot: a bot for automated and interactive domain modelling. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–10.