Substitution Graph Learning: An Implementation

Katie Rischpater
katie.risc@gmail.com

University of California Los Angeles — July 30, 2022

Introduction

This project intends to implement an alternate algorithm for Substitutable Context Free Grammars (CFG). The framework for this algorithm was taken from the research done by done by Alexander Clark and Rémi Eyraud in their 2007 paper *Polynomial Identification in the Limit of Substitutable Context-free Languages*. By linearly parsing a given sentence, and analyzing the context of each sub-string found within, we can generate a *Substitution Graph* (SG) of the sentence. This graph, detailed within Section 3 of Clark and Eyraud's paper, allows a learning algorithm to find sub-strings with a shared context, and create a Zero-Reversible CFG (Clark and Eyraud 2007).

The goal of this project was to expand upon the string parsing capabilities of the Substitution Graph. Rather than analyzing a single string, this program's expands on the learning capabilities by considering an entire body of text. Within this project, Σ^* and Σ retain the same definitions used in lecture, where Σ is the alphabet and Σ^* is all possible strings over thee alphabet. The goal to learn a target language L .

Definition (Substitution Graph) 1 *Given a finite set of sentences S', define a Substitution Graph as follows:*

$$
\begin{aligned}
S' &= \{S' \subseteq L : S' \neq \emptyset \} \\
I &= \{u \in \Sigma^* : \exists\, l, r,\ lur \in s\ ,\ s \in S'\} \\
E &= \{u +\!\!+ v \in \Sigma^* :\ \exists l, r,\ lur,\ lvr, \in s,\ s \in S'\}
\end{aligned}
$$

By summing over each string in S', we can construct a graph from a much larger swath of data. By doing this, we can effectively merge Algorithms 1 and 2 defined in Clark and Eyraud's paper. Rather than merging several CFGs, we simply concatenate the sentences of the input, and add the rules to a hash-table for the final grammar. For the sake of simplicity, CFGs follow the usual grammar definition (Clark and Eyraud 2007),

Demonstration

Within the README.txt provided, there are instructions for how to run the code. The script is run as any other Python script, with the addition of several flags to change the input and output. For example, the user could generate a visual graph without the text output by running the following...

Command Line

```
$ ls
SGL-Learner.py
$ ./SGL-Learner.py -p -v 1>/dev/null
Generating Graph...
Drawing Figure...
Generating Graph...
Drawing Figure...
$ ls
SGL-Learner.py SG_1.png SG_2.png
```

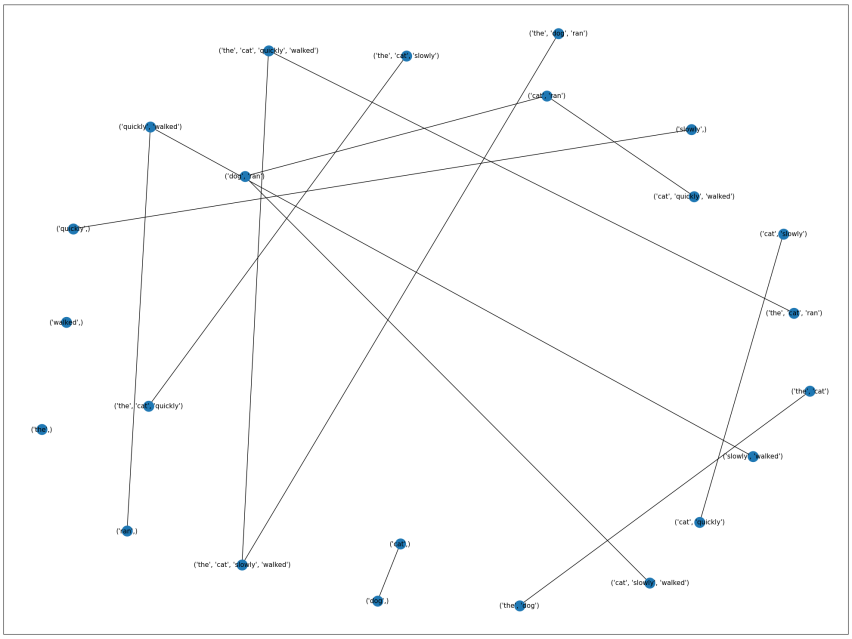

Figure 1: SG_2.png Graph Visualization

Above is an example output of one of the two builtin test cases, SG_2. The –visualize feature requires two python modules, which are detailed in README.txt. For convenience, an additional copy of the script has been included without the –visualize feature, so the program can be ran without installing these modules. For more information on the possible modes of execution, the –help flag can be used.

Implementation & Observations

The Substitution Graphs generated by this script analogous to the Observation Tables that we discussed during lecture. By analyzing the context of each individual sub-string u, we can create equivalence classes $[u]$ for each string that shares a given context l, r. With these classes, we can group together like phrases, and generate a CFG that collapses these distinctions. When run with some rudimentary texts, such as those in the provided sample-language.txt, the resulting CFG will correctly group together constituents. For example, a rule is generated that correctly combines two adverbs.

```
{('slowly',), ('quickly',)} --> ('slowly',)
{('slowly',), ('quickly',)} --> ('quickly',)
```

Throughout this project, I found this terminal rules to be the most fascinating. Once the algorithm was correctly implemented, several of these "constituent grouping" rules could be observed. Some, of course, do not represent proper constituents; however, there was an impressive number of correct groupings.

Discussion & Further Development

A CFG that is trained from a Substitution Graph is, unfortunately, not the most practical. As Clark, Eyraud, and several others have discussed, a Substitution-Graph trained language tends to over-generate (Clark and Eyraud 2007). Furthermore, this script was lacking in some of the optimizations required for the

conversion between SG and CFG to be efficient. One of the graphs provided was generated from Edgar Allen Poe's *The Raven* - While visually impressive, the production of this graph took almost 10 minutes.

While an unmodified Substitution-CFG (S-CFG) may be too bulky for practical applications, they still create an interesting avenue for experimentation. As proved by Clark and Eyraud, these grammars are Gold Learnable, and can be altered to run in a more timely fashion (Clark and Eyraud 2007). In the future, I would like to explore a modification of the Substitution Graph - rather than simply recording substrings and their contexts, I think it would be of value to also record the frequency of these sub-strings. By keeping track of both sub-strings and the frequency they appear in a given context, and then somehow converting these observations into a Probabilistic CFG, I think it would be fascinating to see how these algorithms predicted the difficulty of parsing future sentences.

There are still several interesting areas to explore through the lens of Substitution learning. We discussed Zero-Reversible Finite State Automata within lecture, however there are other formalisms that can use this style of learning. In later research, Clark explores substitutability in Multiple-CFGs - it would be interesting to try and adapt this script to MCFGs, as was done in this paper (Clark and Yoshinaka 2016).

References

Clark, A., & Eyraud, R. (2007). Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, *8*, 1725–1745. https://jmlr.csail.mit.edu/papers/volume8/clark07a/clark07a.pdf

Clark, A., & Yoshinaka, R. (2016). *Topics in grammar inference*. https://link.springer.com/chapter/10.1007/978-3-662-48395-4_6