Babble: Learning Grammar Structure with Generative Adversarial Networks

Brian Clark, CWRU

December 18, 2017

Contents

Abstract

This is the text of the abstract

1 Introduction

1.1 Problem

The English language is undeniably a complex construct. English grammar is composed of dozens of rules, many of which are only loosely applied. The vocabulary is incredibly vast, and growing everyday. English is consistently rated one of the most difficult world languages for new speakers to learn.

And yet, even young children are capable of producing speech that is widely accepted as "English" – at least, most of the time. Clearly, the human brain has a well developed capability for internalizing the deep structure of a language like English and generating examples that match that structure.

This naturally leads to the question: can intelligences other than human brains perform a similar feat? The purpose of this project is to produce a deep learning network that, given examples of text from a language, is able to produce text samples that comply with the rules of that language.

1.2 Grammar

The first challenge built into this learning problem is the definition of a language.

A approach to language definition is to think of a language as a set of formal rules called a grammar [1]. Any piece of text satisfying the rules of the grammar can be considered part of the language inquestion. This grammar can be thought of as an operational definition of a language.

This leads to a natural conclusion in the realm of generative networks: a grammar could serve as a loss function for a network. The network would be rewarded for creating a grammatical sentence and penalized for created a non-grammatical sentence.

The limitations with this approach are the same as the limitations for grammars in general. For any sufficiently complex language, it becomes difficult to write a correct and complete grammar describing the language. For example, there is no one extant grammar that is widely regarded to represent the entirety of the English language. It may not even be possible to create such a grammar.

Without the feedback provided by a formal grammar, there is still one source of grammaticality available to us: actual sentences. Ideally, the generative network could use an existing corpus of sentences in a language as the basis for building a model of the language. Using data itself to glean structure rather than a set of predesigned rules is a mainstay of machine learning. This is the approach followed in this project.

1.3 Generative Adversarial Networks

A tool explored in this project is a deep learning model known as Generative Adversarial Networks [3].

The idea behind the generative adversarial approach is to split the network into two subnetworks, each with a distinct role:

- **Discriminator**: The discriminator is trained to recognize the difference between examples that meet certain criteria and examples that do not. For example, the discriminator would be able to tell if a given sentence was grammatical or not in a language.
- **Generator**: This subnetwork is responsible for transforming random noise inputs into samples that meet the criteria being judged for by the discriminator. In this example, the generator would attempt to create phrases out of random noise that follow the rules of a language.

The insight behind this model, however, is the interplay between the two subsystem. The outputs from the generator are fed into the discriminator and labelled as negative examples. This creates a feedback loop where the discriminator gets ever better at distinguishing fake sentences from real sentences which pushes the generator to produce more realistic sentences.

2 Background

???

3 Approach

Description role of generator inputs outputs role of discriminator inputs outputs overall diagram Grammar definition Abstract definition class diagram? Tools nltk Grammars SimpleGrammar description generation process _Grammar English Network definition Tools keras Generator structure diagram Discriminator structure diagram Oracle mode Train discriminator full Use premade data Use disc as an oracle for the generator Helpful to make a good generator

4 Results

Experiments Oracle mode Teach the discriminator fully Size of hidden layer Use premade data Use trained discriminator as oracle Network structure Amount of data

5 Discussion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

6 Conclusions

Some text, from [2]

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[2] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[3] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks. 2014.