Natural Language Processing Final Project Report

Topic: Grammar Checker

CS 6320.001

Submitted by:

Paul Geevargheese Mathew(pxm142330)

MS in Computer Science

Erik Jonsson School of Engineering & Computer Science

The University of Texas at Dallas

1. Abstract

There have been many unsupervised learning algorithms defined for statistical models for English grammar but their computational complexity makes applying them to a large data sets intractable. This project presents a probabilistic model of English grammar that is much simpler than the other conventional model.

The model which I have used in this project is mainly based on grammatical bigrams by using syntactic relationships between pair of word and their POS tag .I present the result of the project that quantify the representational adequacy of the grammatical bigram model and its ability to induce syntactic structure from large amount of raw data. One important thing to be considered is that grammar checking is very vast even for a declarative statement and here I ‘am considering lots of constraints when I’m trying to do the problem and try to solve it.

Here I’ll have a simple declarative statement as input. First the syntactic structure of sentence is checked with rules specified through the XML file and also with the corpus file. If the sentence has wrong grammar, it will use HMM techniques to find the correct grammar by using Word Likelihood probability and Tag Transition probability.

1. Introduction

This project focus on verifying text with grammatical correctness within the given declarative sentence. Here we will enter a simple declarative statement and it will print whether the syntactic structure is correct or not. For wrong syntactic structure it will predict the correct syntactic structure and correct sentence based on the corpus .There are many business tools like Microsoft word which use these kind of capabilities. I developed a standalone application based on HMM methodologies. The concept of creating a proper grammar checking tool is very hard .This project makes use of Natural Language Processing concepts. . Here our goal is to achieve correct syntactic structure of the sentence.

Here we are provided with the corpus which contains many declarative statements and an XML file which contains the context free grammar.

For predicting the correct grammar I used two main concepts of Grammatical Bigram Model. They are :-

1. Word Likelihood Probability
2. Tag Transition Probability

# Prior Research and Previous Work

Grammatical Bigrams:

The major part of my research was based on paper based on grammatical bigram which was published by Mark A Paskin from University of California ,Berkeley and also based on Speech and Language Processing. Daniel Jurafsky & James H. Martin .This paper presents a probabilistic model of English grammar which admits an efficient usage of expectation maximization training algorithm. The model is based on grammatical bigram i.e. the syntactic relationship between the pair of words. I also looked in the use of Hidden Markov Model for part of speech tagging.HMM also defined a learning algorithm the Baum-Welch (EM ) algorithm that can be given unlabeled data and find the best mapping of labels to observations.

However when we apply HMM for POS tagging ,we don’t use the Baum Welch algorithm for learning the HMM parameter. Instead HMMs for part-of-speech tagging are trained on a fully labeled dataset—a set of sentences with each word annotated with a part-of-speech tag—setting parameters by maximum likelihood estimates on this training data.

HMM decoding should choose the tag sequence that is most probable given the observation sequence of n words.

C:\NLP\1.PNG

When we apply Baye’s rule we get

C:\NLP\2.PNG

If we simplify the equation by removing the denominator P() we get

C:\NLP\3.PNG

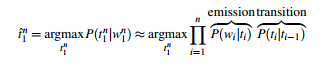
Here HMM taggers assume that the probability of the word appearing depends only on its own tag and is independent of neighboring words and tag .

C:\NLP\4.PNG

Here we make a bigram assumption that the probability of a tag is dependent on the previous tag ,rather than on entire tag sequence.

C:\NLP\5.PNG

Combining all this results in the following equation for the most probable tag sequence from a bigram tagger , which corresponds to emission probability and transition probability from the HMM.



5. Proposed Solution

I’ am implementing the concept of Bigram Model which approximates the probability of a word given all the previous word by the conditional probability of the preceding word.

P(|) ≈P(|)

P( ) ≈ P(|)

Here I applied the same concept on the POS tagging of the sentence which approximates the probability of a POS tag given all the previous POS tag by the conditional probability of the preceding POS tag. Let be the POS tag of the word , then we can say

P(|) ≈P(|)

P( ) ≈ P(|)

So for finding the correct grammar first I created the Bigram model for POS tag which is called Tag Transition and created Tag Transition Probability.

Once I was able to get the correct syntactic structure for the sentence next I will use the concept of Word Likelihood Probability which approximates the probability of a word given the POS tag which is

P(|) =

For each word which I get from the Tag Transition I’ll compare it with the Bigram model for words using Word Likelihood to find the correct word given the correct tag and given preceding word

This can be illustrated with an example.

Suppose I want to check the grammar of sentence “This is are dog”.

1. The corpus is split into sentences

2. Each sentence is split into words

3. Each word is assigned its part-of- speech tag(s)

4. First the program will check the POS Tag with the context free grammar specified in the XML file and also with the tag transition based on the corpus and it will detect whether the syntactic structure of the grammar is wrong or right.

5. If the syntactic structure of the sentence is wrong, it will use Tag Transition and Word Likelihood based on the corpus to find the correct syntactic structure.

So to do this ,first it will check the incorrect syntactic structure in the sentence , for eg “This is are Dog”.

Below shows the POS tagging of the sentence.

DT VBZ VBP NN

This is are dog

Here the syntactic structure is **DT -> VBZ-> VBP-> NN**. Here the incorrect tag transition is **VBZ ->VBP** .

Next it will find the correct tag transition based on the corpus by comparing the tag with the previous tag and the next tag. The correct tag transition will look like **DT -> VBZ -> DT -> NN** (for “This is a dog”).

It will detect that the correct tag transition is P(DT| VBZ) and P(NN|DT) exist based on the corpus.

Now based on the correct POS tag it will try to find the correct word based on Word Likelihood Probability. Here the based on Word Likelihood probability from corpus, P(a|DT) is present .

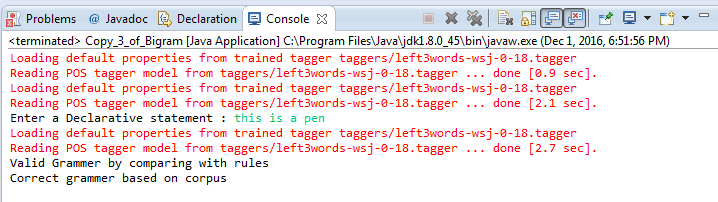
DT VBZ DT NN

This is a dog

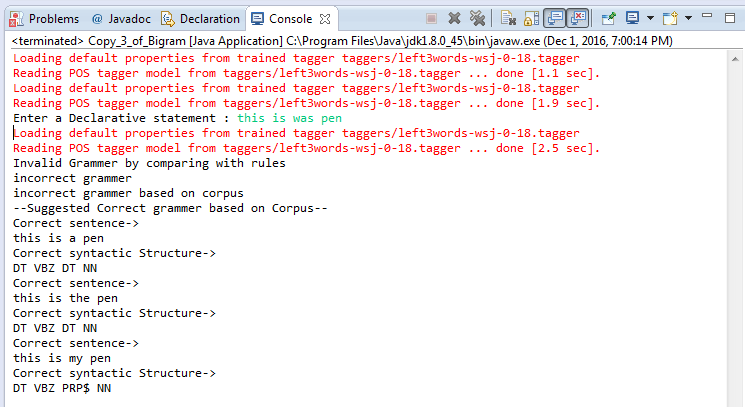
Once we got **DT -> a** ,it will check the corpus for the presence of **is - > a** bigram from the corpus to increase the accuracy. If it is present then the correct sentence is printed as output.

6. Input and Output

First I entered a correct grammatical sentence “this is a pen” , it will show that the grammar is correct based on Corpus and rule.



Next I entered a incorrect syntactic statement “this is was pen”. The output will suggest correct grammar based on corpus



6. Data Used

I created my own Corpus which contains some declarative statement and also has an XML file which contains the grammar.

7. Resources

I used Stanford-postagger.jar for POS tagging of the sentence.

8. Summary

Tag transition probability and Word Likelihood probability represents an exciting new approach in finding the accuracy of syntactic structure and predicting the correct grammar.

I faced certain difficulties while doing the project .I had to constraint lots of constraint while checking the grammar for the sentence.

1. To check the syntactic structure of sentence , the corpus also should have the same structure .
2. If you omit words from the input sentence and want to check the syntactic structure, chances are that it won’t produce correct Syntactic structure
3. Unable to find syntactic structure if the input sentence has unknown word.
4. As a word can have multiple POS tagging ,sometime the syntactic structure of the sentence will be correct not the semantic structure ,thus correct syntactic structure doesn’t mean correct semantic structure or correct meaning..
5. Supervised method fail for the word not in training data.
6. Based on the variation of different sentences , we have to add rules in the XML file for checking the structure.

Performance of this project can be improved by

1. Including larger corpus for different structure
2. Specifying more rules
3. Application of autosuggestion of words if any words in structure is omitted.
4. Viterbi Algorithm can be used to check the correct sense of the word as one word can have multiple POS tagging
5. To achieve higher accuracy, it is also important to have a good model for dealing with unknown words

9.Conclusionand Future Work

Bigram concepts using Tag Transition and Word Likelihood plays a key role in defining the syntactic structure of the sentence. The POS tagging I used for this project is one of many disambiguation task in Natural Language Processing.

I learned that an ideal grammar checking program is very difficult to achieve as the language is vast and it is evolving as the time moves on .Even the best software which performs good syntactic structure examination requires consideration of lots of constraints. It definitely demands more storage space as high performance software requires large corpus.The work done so far is to generate correct syntactic structure of the sentence. But the way to develop a grammar check tool is to implement the relationship model and code the dependencies to detect grammatical errors. Another way to implement the grammar check would be to code the probability parsing score of the sentence and detect if the score is good enough to pass off the sentence as grammatically correct. But this is a crude implementation which might not always be right.

10.References

1. Krishnan, V., S. Das, and S. Chakrabarti. 2005. Enhanced Answer Type Inference from Questions using Sequential Models. The conference on Human Language Technology and Empirical Methods in Natural Language Processing.
2. Grammar Checker Wikipedia (<https://en.wikipedia.org/wiki/Grammar_checker>)
3. Dependency Grammar Wikipedia (<https://en.wikipedia.org/wiki/Dependency_grammar>)
4. Stanford POS Tagger (Software available at <http://nlp.stanford.edu/software/tagger.shtml> )
5. POS Tagging using Stanford Tagger ( <http://new.galalaly.me/2011/05/tagging-text-with-stanford-pos-tagger-in-java-applications/> )
6. Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright c 2016. All rights reserved. Draft of November 7, 2016.