Grammar Checker using Hidden Markov Models and Trained CFGs

Benjamin Brook and Wei Zhang

December 13, 2016

1 Introduction

Parsing natural language has remained a difficult problem in computer science, mainly due to the lack of clarity in the underlying structure of sentences. Words can take on many meanings. For example, in the phrase "I saw her duck", *duck* could be either a verb or a noun. This ambiguity makes it difficult for parsers to determine the meaning of sentences, or the grammatical structure. Our project seeks to model the part-of-speech tag sequences of sentences using Hidden Markov Models, and verify the grammatical validity of input sentences using grammatical constraints defined by a context-free-grammar (CFG).

The Viterbi algorithm as well as Hidden Markov Models are well studied in the course CS182. Context-free-grammars are derived from the theory of computation: in Chomsky hierarchy, Context-Free-Languages, which are generated by CFGs, are the theoretical basis for the phrase structure of most programming languages. They are also an important tool in phrase structure analysis in Natural Language Processing (NLP). Frequently compared to the Naive Bayes Classifier, the Averaged Perceptron classifier is an essential model in Machine Learning, which we fall back to in the case of unrecognized words.

This project does not seek to exhaustively solve grammatical problems in English. Grammatical correctness is simply too abstract and ill-defined. Instead, our project seeks a reasonable approach to model English grammar.

2 Background and Related Work

Natural Language Processing (NLP) is an interesting field in Artificial Intelligence. This field is attractive to both scientists and engineers. Research in NLP is not only reflected in academic papers and journals, but also in open source software projects, such as NLTK, TextBlob, Harvard NLP, Stanford CoreNLP.

Kupiec *et al* demonstrate the use of Hidden Markov Models as part-of-speech taggers [3]. To compensate for incomplete corpora coverage, the team used context and suffix information (we used an averaged perceptron model to compensate). Kadioglu *et al* research grammar-constraint

satisfaction problems and demonstrate the efficacy of context-free-grammars [2]. The team built an arc-consistency algorithm to satisfy context-free-grammars (we did not build a CSP solver, but rather used the precedent set by Kadiologlu *et al* for CFGs as phrase structure rules). Besides these papers, we used two textbooks *Foundations of Statistical Natural Language Processing* [1] and *Natural Language Processing with Python* [4].

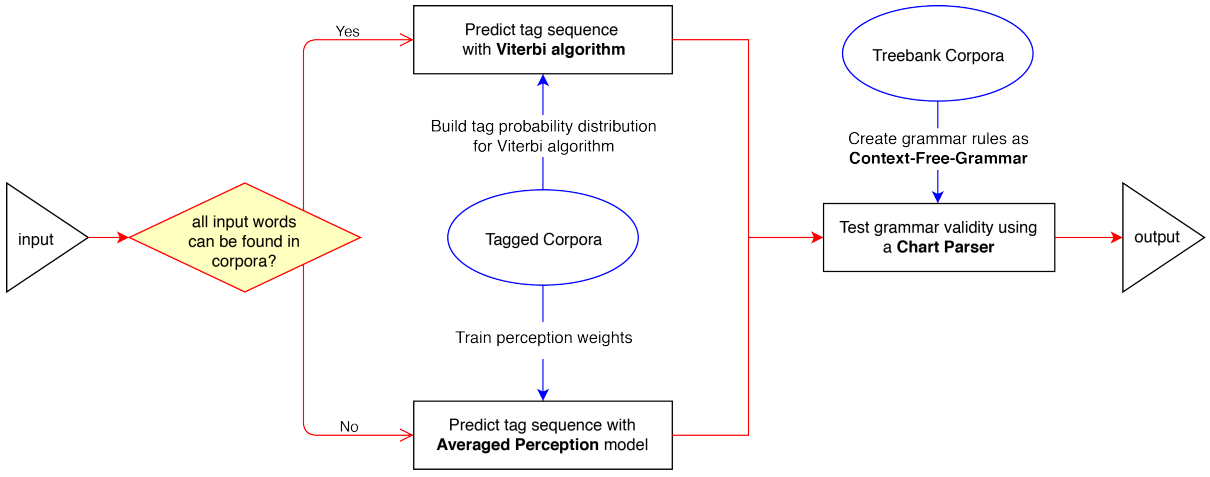
3 Problem Specification

The goal of this paper is to determine whether Hidden Markov Models and Context Free Grammars are an effective way at determining sentence structure and verifying its grammatical correctness. Additionally, we will assess which of the steps (HMM or CFG) bottlenecks the accuracy (if at all) of the grammar checker.

The final product will have two features, a feature where a user can input their own sentences and check its grammatical correctness, and a test feature to check the accuracy of the viterbi algorithm against a corpus. On the input feature, the grammar checker should assign a part of speech tag to each word in the input sentence and then check the grammatical structure. As output, the grammar checker should return the sequence of part-of-speech tags and whether it is grammatically correct according to our model.

4 Approach

A sentence is made up of a sequence of words, each of which represent a grammatical building block. The order of these building blocks is what determines the grammatical validity of the sentence. We call these building blocks ”parts of speech”, and denote them by POS tags. We read a sentence as input, and from this sequence of words we need to find the hidden sequence of POS tags. One can understand this as a Hidden Markov Model, where the Markov state sequence is a sequence of POS tags, and the emission sequence is the sequence of English words. To determine the most likely sequence of POS tags from an input sentence, we used the Viterbi algorithm.



We also used an Averaged Perceptron model as a backup for when a word from the input sentence never showed up in our tagged corpora. It can guess tags for unknown words based on the context surrounding the word. The POS tag sequence for the input sentence is then validated against a context-free-grammar ruleset, which was learned from a treebank corpus. The algorithm validating a sequence against the CFG is the *nltk* ChartParser algorithm. Finally, the program returned its guess of whether the input sentence was grammatically correct.

4.1 Viterbi algorithm

The Viterbi algorithm, as studied in CS182, is a dynamic programming algorithm for finding the most likely sequence of hidden states (the Viterbi path) in a Hidden Markov Model.

Algorithm 1 Decoding: Viterbi Algorithm

$viterbi[s,t]$: the Viterbi path probability in time step t and in state s
a_{ij}: the transition probability from previous state i to current state j
$b_j(o_t)$: the state observation likelihood of the observation symbol o_t given the current state j
procedure VITERBI(observation of len T, state-graph of len N)
  for each state s from 1 to N **do**
    $viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
  end for
  for each time step t from 2 to T **do**
    for each state s from 1 to N **do**
      $vitebi[s,t] \leftarrow \max_{s\prime}^{N}[s\prime = 1, t-1] * a_{s\prime,s} * b_s(o_t)$
      $backpointer[s,t] \leftarrow \arg\max_{s\prime=1}^{N} viterbi[s\prime, t-1] * a_{s\prime,s}$
    end for
    $viterbi[q_F,T] \leftarrow \max s = 1^N[s,T] * a_{s,q_F}$
    $backpointer[q_F,T] \leftarrow \arg\max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$
  end for
  return : the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$
end procedure

4.2 Averaged Perceptron

Averaged Perceptron, as a type of linear classifier, is an algorithm for supervised learning of binary classifiers. This algorithm is employed in the Natural Language Toolkit NLTK3.0. In our project, the tagged corpora are used but the possibility that the input word is out of the tagged corpora always exists.

The same tagged corpora are used to train the Averaged Perceptron tagger. This ensures consistency in the style of POS-tags, as these vary among corpora. Therefore the output of both the Viterbi tagger as well as the AP tagger were consistent before being fed into the CFG.

Algorithm 2 Averaged Perceptron

procedure AVERAGED PERCEPTRON
Initialization: $(w_1, w_2, \ldots, w_n) = 0$
for $i = 1, 2, \ldots$ **do**
Receive : $x_i \in R^d, y_i \in \{+1, -1\}$
Predict : $\widehat{y_i} = \text{sign}\,(w_t \cdot x_t)$
Receive correct lable : $y_i \in \{+1, -1\}$
Update : $w_{i+1} \leftarrow w_i + y_i \cdot x_i$
end for
Output: $w = \sum_{i=1}^{n} \frac{w_i}{n}$
end procedure

5 Experiments

5.1 Results

Of 1500 correctly tagged sentences from the Brown corpus, our Hidden Markov Model and Viterbi algorithm yielded a perfect tag sequence in 70.6% of sentences. In contrast, the Averaged Perceptron model yielded the correct sequence in 90.7% of sentences. The trained CFG and Chart Parser algorithm step was only 45% accurate of 100 input sentences we manually entered.

It should be noted that many grammatical structures on the test set for the context-free-grammar were never seen before in the treebank corpus, causing the CFG to fail frequently in this way.

It should also be noted that the Brown corpus and conll2000 corpus have slightly different tag sets. The Brown corpus tag set is a superset of the conll2000 corpus. It has more specific tags (such as a very particular type of verb). The CFG tagger uses the Treebank corpus, which is the same set as the Brown corpus. This means that some conll2000 tests may have failed due to a prediction with a Brown tag out of the conll2000 vocabulary, or some Brown tests may have failed due to a simplified conll2000 tag being predicted when it was looking for a specific Brown tag. This means the accuracy results are probably understated. This was a necessary tradeoff to increase the vocabulary of our grammar checker. With only one corpus, the tagger step fails often on user input due to words out of the vocabulary.

	Viterbi Score
conll2000	70.6%
Brown	55.0%

Table 1: POS Tagger Accuracy per corpus. Note that Brown uses a much larger list of POS Tags. Additionally, accuracy is a just a measure of the actual POS sequence. Therefore some tag sequences may be counted as inaccurate, despite being a valid interpretation of the sentence itself.

	Averaged Perceptron Score
conll2000	90.7%

Table 2: The averaged perceptron scored very well on the conll2000 set. This paper focuses on evaluating the HMM/viterbi model, so the efficacy of the averaged perceptron model as a general POS-tagging method was not explored exhaustively.

6 Discussion

The Hidden Markov Model and Viterbi algorithm are an effective way of identifying POS tag sequences of sentences. Trained CFGs, however, are a poor method of modeling English. Natural language has too many exceptions to its rule sets, something that CFGs are limited in modeling. The CFG checker was the bottleneck in our pipeline, dropping from 70.6% accuracy at the tag-sequence stage to 45% accuracy at the CFG stage.

In the Viterbi algorithm, a bigram is used as the elements of a 1st order Markov Chain model. If a trigram (2nd order Markov Chain) or greater is employed, the accuracy of HMM may be improved. The Kupiec *et al* paper produces over 90% accuracy [3] through expansion of the n-gram.

English grammar is a very hard problem because it is so ill-defined. Each step in our pipeline seeks to transform or abstract the sequence of words toward our output, but each step comes with noisiness and imperfect modeling. Part of speech tags, for example, are an imperfect generalization for the underlying words. There may be a valid part-of-speech tag sequence that in general is grammatically correct, but in this case, because of the actual underlying words, is not.

In general, grammatical ambiguity will always be a burden on natural language processing. We have shown that Hidden Markov Models and the Viterbi algorithm are an effective way of generalizing words into their part-of-speech tags, and that CFGs are not an ideal way of describing English grammar rules in practice. This is why many grammar checkers today have explicitly defined and hardcoded rules, rather than a single trained model for predicting grammatical correctness.

A System Description

Appendix 1 How to run our program.

Run `python master.py` to use the grammar checker.
This file will structure the HMM using two corpora (function in `viterbi_tagger.py`), build the averaged perceptron model (in `perceptron_tagger.py`), generate the context-free-grammar based on the treebank (in `cfg.py`), and then ask for user input. After the user enters a sentence, the system gets the tag sequence from either the Viterbi algorithm or AP tagger (depending whether all input words exist in the corpora). The tag sequence is then passed into the context-free-grammar where the nltk ChartParser algorithm tests it and returns True/False for the validity of the grammar.

If desired, run `python test.py` to test the viterbi tagger. This also trains and tests the averaged perceptron tagger. It has a long runtime, but produces the results in *Table 1* and *Table 2*.

If desired, run `dataAP/buildTrainTestFiles.py` to generate the .txt to retrain the Averaged Perceptron (AP) tagger. Users can change the training set in `dataAP/train.txt`.

B Group Makeup

Appendix 2

Participants: Benjamin Brook and Wei Zhang

Both of us worked on all aspects of the project, from coding to writing.

Our GitHub repository for the project:
https://github.com/bencmbrook/Grammar-Checker

References

[1] Hinrich Schutze Christopher D.Manning. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 2002.

[2] Serdar Kadioglu and Prof Meinolf Sellmann. Grammar constraints: Combining expressiveness with efficiency. 2009.

[3] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[4] Edward Loper Steven Bird, Ewan KLein. *Natural Language Processing with Python*. O'Reilly, Sebastopol, California, 2002.