Lucas Elbert
lucas.elbert@polytechnique.edu

Algorithms for speech and NLP
TP2

03/08/20

In this project I developed a probabilistic parser for french language that is robust to unknown words based on a PCFG, the Levenshtein distance, word embeddings and a probabilistic CYK-algorithm. The parser is trained on the SEQUOIA database, reads in a sentence and outputs a predicted parse tree.

1 The approach

I provide an implementation for all modules and do not rely on external nlp-focused libraries (only the visualization uses *graphviz* but is not needed for the parsing itself). One of the main components of my implementation are trees with various functionalities to represent parse trees.

1.1 Learning a PCFG

Bracketed expressions $\rightarrow$ trees
To learn a pcfg I first read parse trees in bracketed form and convert them to tree instances by a linear scan of the string and keeping track of opening and closing brackets whilst building up the tree.

Trees $\rightarrow$ Chomsky normal PCFG
A PCFG could easily be extracted from the trees by taking their nodes as symbols (leaves as terminals) and parent-children relations as rules. But to obtain a grammar in Chomsky normal form I preprocess (normalize) each tree by splitting and collapsing of nodes and other operations corresponding to the normalization rules of grammars applied in the tree. Extracting rules, symbols and counts from the normalized trees now directly inferes a PCFG in CNF. Some special attention is paid to split the PCFG in a grammar with POS-tags as terminals and a lexicon from POS-tags to tokens. For numerical stability the probabilities are kept in log-scale.

1.2 OutOfVocabulary

To make the parsing of new sentences robust to typos and unknown words I use the Damerau-Levenshtein distance, embedding similarities and an extension of the CYK algorithm to also include grammatical knowledge in the OOV module. The extension allows to simultaneously consider several correction attempts. When facing an unrecognized input word I first consider it as a typo and consider words from the training corpus with low DL-distance as corrections. Those are found by DL-distance computation for all words passing a fast character count comparison. If no typo-correction is found I try to find the input word in the embedding corpus and consider close known words as representative. Otherwise the word is replaced by an 'UNK' token that can be emitted from any POS-tag.

1.3 CYK - Parsing new sentences

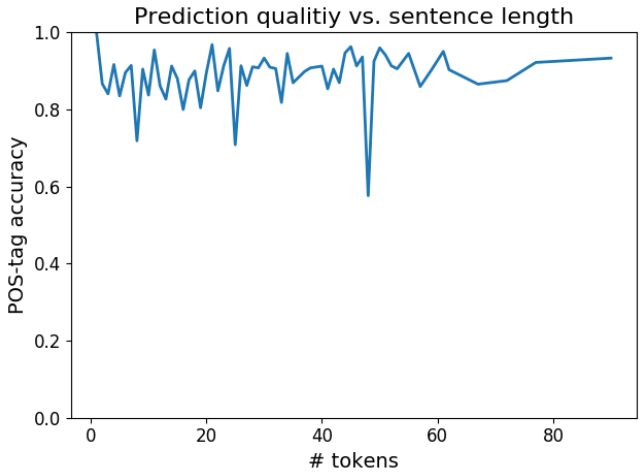
CYK
For tokens $t_0, ..., t_n$ the CYK algorithm constructs a table C such that $C[i, j]$ contains all symbols that can be derived to the subsentence $t_i, ..., t_{i+j}$. The table is constructed by bottom-up dynamic programming. As we are only interested in the most probable tree we store in each cell for each symbol only the derivation with highest probability. To be able to reconstruct the derivation I directly store trees in the table's cells and link them to each other (only small overhead). As an extension I allow the CYK algorithm to take as input a sequence of token sets $T_1, T_2, ...T_n$ and the cell $C[i, j]$ contains symbols that have derivation $d_1, d_2, ..., d_n$ with $d_i \in T_i \ \forall i$. To optimize the runtime I implemented two ways of computing a cell $C[i, j]$ and dynamically choose between them.

Tree normalization inversion
Finally for a better readability I 'inverse normalize' the tree by undoing the binarization and nonsolitary

terminal removal. Only the removal of unitary rules can not be reversed as the information was discarded.

2 Results and improvements

The system was trained on the first 80% of the SEQUOIA database and tested on the last 10%, making 309 sentences. The system was able to parse 306/309 sentences and it found for 87.7% of all test tokens the correct POS-tag. The opposite table shows the size of the learned PCFG.
To see whether the system's quality is influenced by the sentence length I measured the POS-tag prediction accuracy dependent on the sentence's length. As the opposite graph shows, the accuracy seems not to be influenced by the length of the sentence. Of course this did not compare the complete tree structure which would need a separate analysis. From manual inspection there seems to be slightly more variation in the tree structure for larger trees.



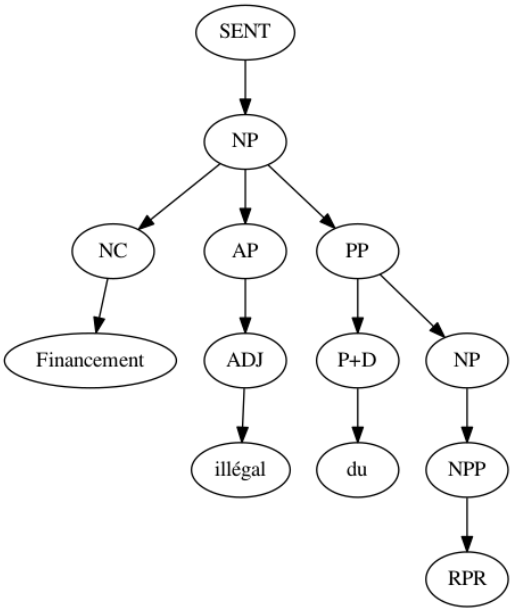
	size
Grammar rules	6370
Lexicon rules	9434
Non-terminals	3463
POS-tags	29
Vocabulary	8958

Table: PCFG size

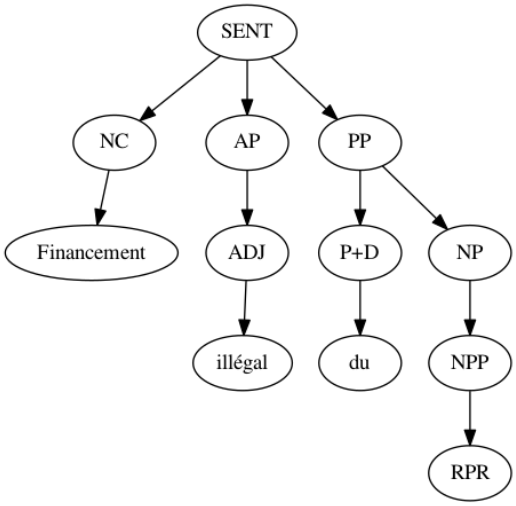
An improvement of the OOV module might be possible by directly combining the Levenshtein distance and the embedding similarity. For an unknown word with several corpus words of close Levenshtein distance we could weight the corpus words by a combination of the Levenshtein distance and cosine similarity to a sentence embedding.
Furthermore we could also improve the runtime by parallelizing the CYK algorithm.

A parsing example with input: 'Financement illégal de RPR'

Ground truth tree

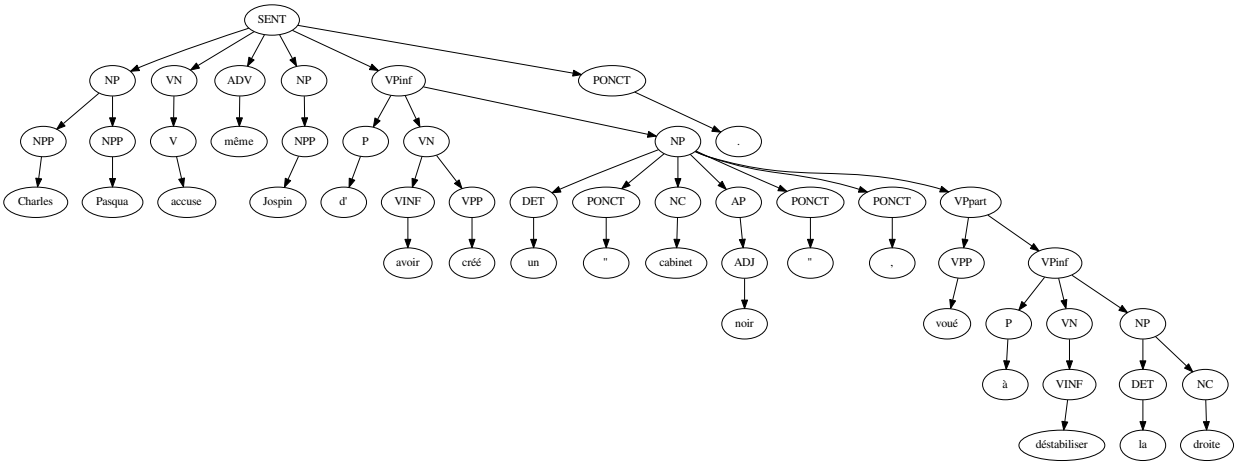


Predicted tree



3 Appendix

Ground truth tree



Predicted tree