Algorithms for speech and natural language processing

Report

TP2 - Probabilistic Parser for French

Student
Mehdi Boubnan

Professors
Benoît Sagot
Emmanuel Dupoux

école normale supérieure paris–saclay

1 Building the parser

We will build a basic probabilistic parser for French that is based on the CYK algorithm and the PCFG model, and that is robust to unknown words thanks to a module for handling out-of-vocabulary words (OOVs), based on levenstein edit distance and word embedding similarity. Theses modules will be applied on the SEQUOIA treebank v6.0, that we split into three parts ignoring the functionnal labels : a training corpus (80%), a validation corpus (10%), and finally a testing corpus (10%).

1.1 Grammar extraction

We begin by extracting the PCFG from the training corpus, that will be used by the CYK algorithm to parse a sentence. Therefore, the Grammar must be in Chomsky Normal Form. To do so, we use the NLTK package to process the provided parse trees removing empty brackets, then convert them to their Chomsky Normal Form (CNF), collapsing all the unary productions. This process requires the creation of many "artificial" non-terminal nodes. With Horizontal Markovization (n=2), we shorten some of these annotations. This leads to creating 12 start symbols in our grammar (due to ROOT collapsing and removing empty brackets), all beginning with `SENT`; we will explain how we will deal with this in the CYK section. The conditional probabilities (right hand side of a rule given a non terminal) are computed by normalizing the counts of these rules occurrences.

1.2 OOV module

To deal with unseen words, we use an OOV module that assigns a (unique) part-of-speech to any token not included in the lexicon extracted from the training corpus. The OOV have to propose similar words based on the combination of the Levenstein distance and an embedding similarity using the Polyglot embedding lexicon. We opted for the normal Levenstein distance instead of the Damereau-Levenstein one due to computational time (the damereau can be used in the system proposed when activated through the arguments). For the testing corpus, we noticed that the majority of out-of-vocabulary words are not spelling errors, but words that are completely out of the vocabulary. Many of the spelling errors are within an edit distance of one (changing an uppercase with a lowercase or changing an accentuated letter with a normal one for example). The idea is then to propose more words in our vocabulary from the embedding based similarity than words from the formal based similarity. We take the n_1 most similar words based on the embedding and n_2 most similar words based on the Levenstein distance. When more than n_1 (resp. n_2) have the same similarity, we take the first n_1 (resp. n_2) from the alphabetically ordered proposed words. Tuning these two parameters give us : $n_1 = 20$ and $n_2 = 2$. We then use a bigram language model with linear interpolation with unigram model to avoid unseen bigrams, to finally choose the most probable word.

$$P_{li}(w_k|w_{k-1}) = \lambda P_{uni}(w_k) + (1-\lambda)P_{mle}(w_k|w_{k-1})$$

We take $\lambda = 0.2$. We use the logarithm of the probabilities to avoid vanishing scores.

1.3 CYK Parser

The algorithm

The parsing will be done using the CYK algorithm, which is in fact an algorithm designed to solve membership problems. It is a bottom-up algorithm, based on dynamic programming, that checks if a sentence belongs to a given CFG in its Chomsky Normal Form. The CYK can be converted to a parsing algorithm, by keeping track of the rules, with the highest conditional probabilities, that constructed the sentence, beginning from the words (bottom of the tree). When we end up with a starting symbol, the sentence belongs then to our Grammar and we can therefore build the tree recursively using the backtrack from this starting symbol. The binarisation of trees we did with NLTK also collapses the ROOT, which makes us having in fact 12 "collapsed starting symbol", we have to check then if we end up with one of these axioms and backtrack from the one having the highest probability. If not, we return a parse that indicates that the sentence doesn't belong to our Grammar. Finnaly, we reverse with NLTK the CNF ("unchomsky") of the constructed parse tree and return it.

Optimization of the algorithm

We apply the CYK parser on the PCFG extracted from the training corpus, where we have only binary productions and the probabilistic lexicon having unary rules `POS` $\rightarrow$ `word`. After looping on the POS tags as a first step, looping on all the possibilities of the binary rules `A` $\rightarrow$ `BC` in the algorithm from the bottom to the top take a lot of time (many hours to parse the whole testing corpus). To improve that, we create two empty sets Set_{left} and Set_{right}. The first one will contain only non-terminals `B` whose score was updated and that appear at least once in the left part of a binary rule right hand side. Similarly, the other set is for the right part non-terminals `C`. Each time we update a score of a non terminal that appear in the right hand side of a binary rule, we add it to the corresponding set. The loop will be then only on the cartesian product of these two sets intersected with all the binaries of the grammar. This trick make us able to parse all the test corpus in **16 minutes** with a single processor, and **5 minutes** when multiprocessing with 8 processors (the system proposed can activate multiprocessing through its arguments).

2 Evaluation and error analysis

To evaluate our modules, we use the evalb bracket scoring metric provided by the package PYEVALB. After extracting the PCFG on the first 90% of the SEQUOIA corpus, we run our parser on the last 10% of the corpus. With the parameters $\lambda = 0.2$, $n_1 = 20$, and $n_2 = 2$, we obtain the followings results :

Parse Recall	Parse Precision	POS Recall	POS Precision
61.77%	58.39%	77%	86.77%

We also find 41 sentences out of the grammar, which is due to our OOV that propose inadequate words leading to a out of grammar parses. Indeed, when extracting the PCFG from the whole corpus, all the sentences are in our grammar and the POS Recall raises to (95.26%). The OOV module must be improved.

Our OOV is built to propose a new word in a greedy way, we can improve that using beam-search or viterbi algorithm to return the most probable sequence with the words of our vocabulary, which can lower out of grammar sentences. Moreover, if the most probable proposed sentence by the OOV give an out of grammar parse, we can try parsing the next proposed sentence (in terms of probability) by the OOV and so on until finding a sentence that belongs to our grammar. We didn't implement this idea since it will take more time to parse all the sentences.

In addition to that, we keep only track of the binary rules that constructed a sentence in the CYK algorithm. Therefore, for sentences having only one word, we don't have a backtracking. In this case, we begin with the axiom having the highest probability as the node of the parse tree, with the word as a child, then we "unchomsky" the parse as explained before.