MVA - Algorithms for Speech and NLP TD 2

Dimitri Meunier

1 Implementation details

The output of the project is a probabilistic parser that takes as input a sentence with potentially Out Of Vocabulary words and outputs the most probable parse tree. It is composed of three modules: a PCFG that allows assigning probability to each parse tree, a PCYK that extracts the most likely tree from raw tokens, and an OOV module that deals with OOV words.

1.1 PCFG-Lexicon

A PCFG [1], [2], assigns a probability to each parse tree T of a sentence S which is defined as the product of the probabilities of all the n rules used to expand each of the n non-terminal nodes in the parse tree T. Each rule i can be expressed as $\alpha_i \mapsto \beta_i$, and the probability of T is:

$$p(T) = \prod_{i=1}^{n} q(\alpha_i \mapsto \beta_i)$$

q can be estimated with the maximum-likelihood estimator $q_{ML}(\alpha \mapsto \beta) = \frac{\text{Count}(\alpha \mapsto \beta)}{\text{Count}(\alpha)}$

Remark. In this project, it was asked to build the PCFG using the terminal POS as terminal tokens instead of the words. In the PCYK, we glue back the sentences with the PCFG using a lexicon that assigns to each word a probability distribution over the terminal POS.
Since the PCYK runs on parse trees in Chomsky Normal Form, we convert each train parse sentence to Chomsky Normal Form on the fly during training an extract the rules from the modified tree. As a result, our PCFG is a well-defined Chomsky Normal Form probabilistic grammar.

1.2 PCYK

Taking inspiration from the pseudo-codes in [1] and at Wikipedia CYK, we build the PCYK from the PCFG and the lexicon. The input is a new sentence in raw tokens and the output is the most likely parse tree (in bracketed format) that can yield to this sentence. If the input sentence is S, the output is:

$$\hat{T} = \arg\max_{T \mapsto S} p(T)$$

On the contrary to a deterministic CYK that can only say if a sentence can be derived from a given grammar, the PCYK can output the best tree that can lead to the sentence.

Remark. Since the PCYK works with a Chomsky Normal Form grammar, the outputs are also CMF trees. We added a post-processing step to invert the CMF transforms and to compare the outputs with the true labels.

1.3 OOV

When testing the parser, there are lots of words that are not included in the training set vocabulary, the OOV words. Using the (Damerau-)Levenshtein distance, polyglot word embeddings [3] and a language model, the OOV module suggests words to replace the OOV words.
From a sentence S, we used the Damerau-Levenshtein distance (instead of the raw Levenshtein distance as it leads to better results) to extract n_1 nearest neighbors for each OOV words in S. On the other hand, we used the polyglot embeddings to extract n_2 nearest neighbors. It leads to $n = n_1 + n_2$ candidates to replace

each OOV word. Let us note that some misspelled words are not included in the polyglot vocabulary, for those words we only use the Levenshtein neighbors.
To choose the best neighbors among the n candidates we use a language model. Similarly to the first homework, the language model includes the bigram and unigram from the training corpus. We add a linear interpolation smoothing to avoid unseen bigram bias, leading to,

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

Since the probability of a sentence can be approximated with $\prod_{k=1}^{|S|} P_{smooth}(w_k|w_{k-1})$, we pick the set of OOV candidates that maximizes the probability of the modified sentence.
We tuned n_1, n_2 and λ using the validation set and found $n_1 = 2$, $n_2 = 20$ and $\lambda = 0.2$.

1.4 Training details

The corpus was split between a training set (80%), a validation set (10%) and a test set (10%). We used the validation set to tune the hyperparameters, then, we train again the system on the train set plus the validation set. For evaluation on the test set, we used the Python multiprocessing library. It runs under 70 minutes on an Intel Core i9-9900K Processor. We are however conscious that many parts in the code can be coded more cleverly to speed things up. The main bottleneck is the PCYK as its complexity is cubic in the length of a sentence.

2 Results

Out of the 309 sentences in the test set, 85 could not be parsed. We used pyevalb to compute the scores from the parsed sentences and obtain 88.89% of tag accuracy. Figure 1 presents the evolution of the scores with respect to the length of the input sentence and the number of non-parsed sentences with respect to the size of each sentence. A striking conclusion is that the longer the sentence the easiest it is to parse. The algorithm struggles a lot with sentences of length inferior to five.
The large number of non-parsed sentences can be explained by the poor performance of the OOV module. This could be improved with a larger training set and a better way to handle misspelled words. Furthermore, the OOV module does not handle true-word errors which further reduces performances.
Another flaw is that the similarity captured is not necessarily correlated to a POS similarity. For example, the Levenstein similarity will replace "beau" by "eau", the first one being an adjective and the second a noun. The embedding similarity seems however to better capture POS similarity.

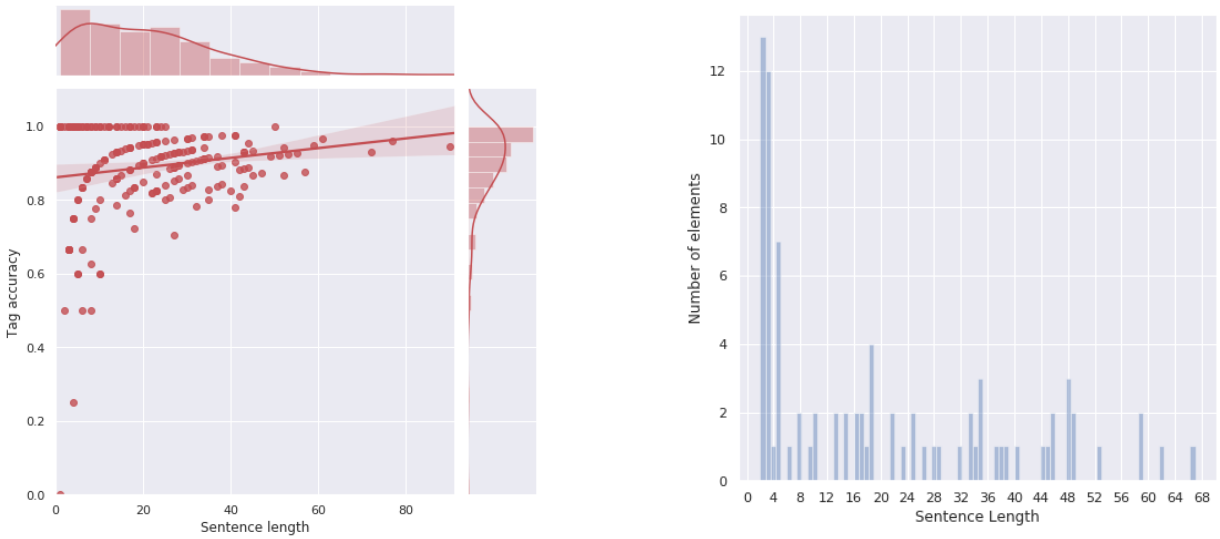

Figure 1: (*left*) Joint plot of the evolution of the tag accuracy with respect to sentence length. (*right*) Bar plot of the number of non-parsed sentences with respect to sentence length.

References

[1] Michael Collins. "Lexicalized probabilistic context-free grammars". In: *Lecture Notes* (2013).

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2009. ISBN: 0131873210.

[3] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. "Polyglot: Distributed Word Representations for Multilingual NLP". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 183–192. URL: http://www.aclweb.org/anthology/W13-3520.