MVA 2020 - Speech and NLP

Ariane ALIX

March 9th, 2020

---

1 Description of my implementation

Apart from the 3 main modules that we will discuss below, I also implemented 2 functions to process the data: one to build clean *nltk* trees in Chomsky normal from, and one to get clean sentences without tags for testing.

1.1 Probabilistic Context-Free Grammar

The PCFG module is composed of three main functions:

- *get_build_grammar*: uses the Chomsky trees of the training sentences and extracts their production rules to build an *nltk* Grammar (using the clean and easy way: *nltk*'s *induce_pcfg*).
- *get_build_lexicon*: iterates throughout all the leaves of all the trees to count the tag-token occurrences. It also looks at tags of higher levels if they are only associated with the token; e.g with '(NP (NPP Gutenberg))' it will count 1 tag 'NP+NPP'.
- *get_build_tags_dicts*: returns dictionaries to easily switch from an index of tag to the corresponding string and vice versa.

1.2 Out-Of-Vocabulary

This module is responsible for finding a tag for words that we do not know (not among words of the training sentences).

The basic principle when we find an unknown word is:

- If the word is in the Polyglot embeddings vocabulary:
    1. Get the 5 most similar word of the training corpus using cosine similarity.
    2. Using those cosine score as weights, compute the weighted average probability of each tag.
    3. Return tag with highest probability.
- If the word is not in the Polyglot embeddings vocabulary:
    1. Find the 5 closest spelled word of the Polyglot vocabulary using the Levenshtein distance.

2. If those words are not in the training vocabulary, find the most similar one using cosine similarity.
3. Return tag using **2.** and **3.** of previous case.

1.3 Probabilistic CYK

There are 2 main functions in the CYK module:

- *parser*: build two tables (*values* and *backtracking*) storing the probabilities of all the potential decomposition of the sentence regarding substrings and the grammatical rules.
- *decode_result*: uses the backtracking table to reconstruct the most probable decomposition of the sentence by choosing the probable best split indices and rules recursively along left and right sub-sentences.

2 Results and ideas

2.1 Results

Example of results obtained when launching the Bash script:

```
ariane@MSI:/mnt/d/Docs/Cours MVA/Speech and NLP/TP2$ ~/run.sh
Name of input file with sentences to parse ? e.g file.txt
file.txt
Name of output file to store the results ? e.g output.txt
result.txt
Loading Polyglot embeddings...
Nb of words: 100004
Length of the embeddings: 64

Loading and processing the SEQUOIA corpus...
Time taken: 0.109 s

Generating Probabilistic Context-Free Grammar from training data...
Lexicon shape (nb of tags, nb of tokens): (56, 8958)
Time taken: 36.643 s

Parsing example from training set: 'Cette exposition nous apprend que dès le XIIe siècle , à Dammarie-sur-Saulx , entre autres sites ,
 une industrie métallurgique existait .'
Result: ( (SENT (NP (DET Cette) (NC exposition)) (VN (CLS nous) (V apprend)) (Ssub (CS que) (PP (P dès) (NP (DET le) (ADJ XIIe) (NC si
ècle))) (PONCT ,) (PP (P à) (NP (NPP Dammarie-sur-Saulx))) (PONCT ,) (PP (P entre) (NP (ADJ autres) (NC sites))) (PONCT ,) (NP (DET un
e) (NC industrie) (AP (ADJ métallurgique))) (VN (V existait))) (PONCT .)))

Parsing invented example: 'Gutenberg est dans les champs'
Result: ( (SENT (NP (NPP Gutenberg)) (VN (V est)) (PP (P dans) (NP (DET les) (NC champs)))))

Loading and processing the sentences from the input file...
Results saved to result.txt
```

Figure 1: Two example sentences and running of Bash script

We launched the parser on the test dataset (last 310 sentences of the corpus). We could not evaluate the accuracy due to lack of time, but the results looks generally good and only 4 of the sentences could not be parsed.

2.2 Potential improvements

A common error we noticed was in the case of a rule that did not exist in the Grammar. It could probably be improved by computing probabilities for the unknown rules, looking at similarity between productions rules for example.

Another error that appears often is for unknown proper noun, since Levenshtein returns the closest spelled word which is not necessarily a proper noun too. This issue could be solved by adding a vocabulary of proper nouns all associated with a 'NP' tag.