Natural Language Processing - Ex3

Please submit a single zip file.
The zip file should contain your code and result files, a README txt file and a single pdf file for the theoretical questions

Due: 31.12.17 23:55

1. (20 pts) Consider a bi-gram Conditional Random Field model:

$$p(y_1 \cdots y_N | x_1 \cdots x_N) = \frac{\prod_{j=1}^{N} \mathrm{e}^{w \cdot f(y_{j-1}, x_1 \cdots x_N, j, y_j)}}{Z(x_1 \cdots x_N; w)}$$

assume $y_1 \cdots y_N$ may take values in the set $\mathcal{Y}$.

(a) Write pseudo-code for a procedure that, given a feature function f, a weight vector (w), an input sentence $x_1 \cdots x_N$, and an index i, outputs the probability distribution

$$p(y_i | y_{i-1}, x_1 \cdots x_N)$$

for every value of $y_i, y_{i-1} \in \mathcal{Y}$.

(b) Write pseudo-code for a procedure that, given a feature function f, a weight vector (w), an input sentence $x_1 \cdots x_N$, and an index i, outputs the probability distribution

$$p(y_i | x_1 \cdots x_N)$$

for every value of $y_i \in \mathcal{Y}$.

2. (80 pts) In this Python programming exercise, we will implement the MST (Maximum Spanning Tree) parser for *unlabeled* dependency parsing, using the perceptron algorithm.

(a) Use the NLTK toolkit for importing the *dependeny_treebank* (using the commands: *nltk.download()* and *from nltk.corpus import dependency_treebank*). Load the parsed sentences of the corpus (given by dependency_treebank.parsed_sents()). Then, divide the obtained corpus into training set and test set such that the test set is formed by the last 10% of the sentences.

(b) **The feature function:**
Assume the input sentence is $s = \{w_1, ..., w_n\} \in S$ (S is the set of possible sentences), so the nodes of the parse tree are $V = \{w_1, ..., w_n, ROOT\}$. Write a Boolean feature function $f : V^2 \times S \rightarrow \{0,1\}^d$ that encodes the following features:

- **Word bigrams:** For a potential edge between the nodes $u, v \in V$, the feature function will have a feature for every pair of word forms (types) w, w', which has a value of 1 if the node u is the word w and the node v is the word w'.
- **POS bigrams:** For a potential edge between the nodes $u, v \in V$, the feature function will have a feature for every pair of POS tags t, t', which has a value of 1 if the node u has the POS tag t and the node v has the POS tag t'.

Remark: The *ROOT* node can be assumed to have the POS tag *ROOT*.

(c) **The perceptron algorithm:**
The scoring function is defined to be the dot product of the feature function by a weight vector w. Implement the averaged perceptron algorithm for learning w from the training set. Use 2 iterations (i.e., two traversals over the examples) and a learning rate equal to 1. Traverse the training instances in a random order to avoid artefacts.

For Inference (computing the MST), use the Chu-Liu-Edmonds algorithm. You can use the following code:

`https://tinyurl.com/ybdoydkl`

For the feature function components based on POS tags, use the part of speech tags given in the test set (no need to run a PoS tagger).

(d) **Evaluation:** Compute the (unlabeled) attachment score for the learned w (i.e., the number of edges shared by the predicted tree and the gold standard tree divided by the number of words; see lecture notes). Report your results in the pdf file.

(e) **Distance features:** Augment the feature function so that it has another feature, such that given $u, v \in V$, has a value of 1 if the node u immediately precedes the node v in the sentence, 0 otherwise. Add similar features for the case where there is one word between u and v, where there are two words between them, and where there are three or more words between them.

Repeat questions (b),(c),(d) using the augmented feature function.