Natural Language Processing - Ex4

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: 27.1.17 23:55

1. (10 pts) Consider the following CFG. Assume that S is the start symbol, and that DT, NNP, NN, VB, IN are POS tags.

   S $\rightarrow$ NP VP
   NP $\rightarrow$ DT NN
   NP $\rightarrow$ NNP
   NP $\rightarrow$ NN
   VP $\rightarrow$ VB NP PP
   PP $\rightarrow$ IN NP
   DT $\rightarrow$ the
   NNP $\rightarrow$ John
   NN $\rightarrow$ crocodile
   NN $\rightarrow$ telescope
   VB $\rightarrow$ saw
   IN $\rightarrow$ with

   (a) Given the presented CFG, draw a parse tree for the sentence:
   "John saw the crocodile with the telescope".

   (b) How many different parse trees are possible for the above sentence under this CFG?

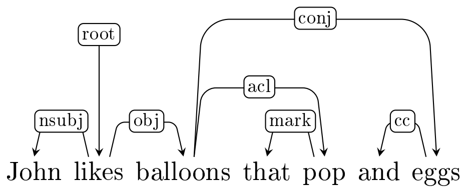
   (c) Now assume that we add head-words to the non-terminal nodes in the parse tree. This is done by specifying the following rules (head-finding rules) for finding the head for each context-free rule:

   - For the rule S $\rightarrow$ NP VP, the VP is the head of the rule.
   - For the rule NP $\rightarrow$ DT NN, the NN is the head of the rule.
   - For the rule NP $\rightarrow$ NNP, the NNP is the head of the rule.
   - For the rule NP $\rightarrow$ NN, the NN is the head of the rule.
   - For the rule VP $\rightarrow$ VB NP PP, the VB is the head of the rule.
   - For the rule PP $\rightarrow$ IN NP, the NP is the head of the rule.
   - For any rule of the form X $\rightarrow$ w, where X is a non-terminal and w is a word, w is the head of the rule. (Then w is the head-word of X).

   Using these head-finding rules, draw a new version of the parse tree you drew in (a), where non-terminals in the tree appear together with their head-word (e.g., VP$_{saw}$). The obtained tree will be a lexicalized parse tree.

   (d) Using the tree obtained in (c), draw a corresponding unlabeled dependency tree for the sentence.

2. (10 pts) Consider the sentence s: "John likes balloons that pop and eggs", and its dependency parse tree below.



   (a) Write a sequence of transitions under the Arc-standard transition system that yields the above tree for s. What is the length of the sequence?

   (b) Write a sequence of transitions under the Arc-eager transition system that yields the above tree for s. What is the length of the sequence?

3. (80 pts) In this Python exercise we will implement a simple open information extraction system that takes text from Wikipedia and extracts triplets of (Subject, Relation, Object), where each of them is a span of text. In this exercise, Subject and Object are names (proper nouns), and Relation is a verb or a verb along with a preposition.

   For instance, given the sentence "Brad Pitt married Angelina Jolie" we will aim to extract the triplet ("Brad Pitt","married","Angelina Jolie").

   Required Packages: In order to carry out the exercise, you will need to install the *spacy* package for NLP and the *wikipedia* package that provides a Python interface for Wikipedia.

   wikipedia can be installed through pip or by directly loading the files:

   `https://pypi.python.org/pypi/wikipedia/`

   See how to install *spacy* here:

   `https://spacy.io/usage/`

   You will also need the default English model of SpaCy called *en*. See instructions here:

   `https://spacy.io/usage/models`

   Once installed, processing a Wikipedia page through SpaCy is straightforward. Example code:

```
import wikipedia, spacy

nlp_model = spacy.load('en')
page = wikipedia.page('Brad Pitt').content
analyzed_page = nlp_model(page)
```

   You can find more details on the *spacy* website, where the two most important data structures are *doc* (for a document) and *token*:

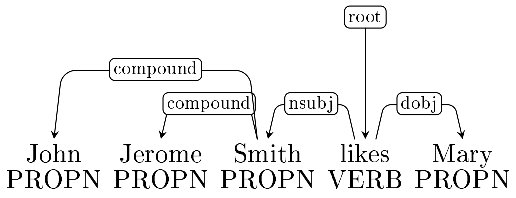
   `https://spacy.io/api/doc`
   `https://spacy.io/api/token`

   (a) Implement an extractor based only on the POS tags of the tokens in the document. The extractor should follow the following steps:

   - Find all proper nouns in the corpus by locating consecutive sequences of tokens with the POS `PROPN`.

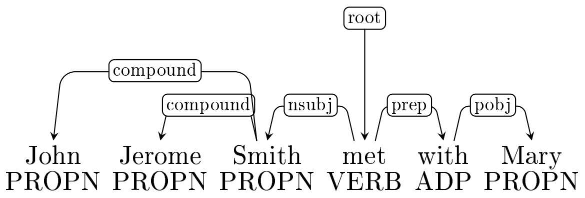
- Find all pairs of proper nouns such that all the tokens between them are non-punctuation (do not have the POS tag PUNCT) and at least one of the tokens between them is a verb (has the POS VERB).
- Upon detecting a pair of proper nouns as above, output a (Subject, Relation, Object) triplet where the first proper noun is the Subject, the second proper noun is the Object, and the Relation is the sequence of tokens between the proper nouns, excluding all tokens which are not verbs or prepositions (i.e., only tokens with POS tags VERB or ADP should be included in Relation).

(b) Implement an extractor based on the dependency tree as well. The extractor should follow the following steps.



- Find all tokens that serve as heads of proper nouns in the corpus (henceforth, *proper noun heads*). Do so by locating all tokens with the POS PROPN that don't have the dependency label *compound* (i.e., the edge from them to their headword is not labeled *compound*). For instance, in the tree above, "Smith" and "Mary" are such tokens.
- For each proper noun head t, define the corresponding proper noun as a set including t along with all its children tokens that have a dependency label *compound* (i.e., the edge from them to t is labeled *compound*). For instance, in the tree above, {"John","Jerome","Smith"} and {"Mary"} are such sets.
- For each pair of proper noun heads h_1 and h_2, extract a (Subject, Relation, Object) triplet if one of the following conditions hold:
  i. **Condition #1**: h_1 and h_2 have the same head token h, the edge (h, h_1) is labeled *nsubj* (nominal subject), and the edge (h, h_2) is labeled *dobj* (direct object).
  ii. **Condition #2**: h_1's parent in the dependency tree (h) is the same as h_2's grandparent (denote h_2's parent with h'), the edge (h, h_1) is labeled *nsubj* (nominal subject), the edge (h, h') is labeled *prep* (preposition), and the edge (h', h_2) is labeled *pobj* (prepositional object).

  An example of proper nouns that meet condition #1 are {"John","Jerome","Smith"} and {"Mary"} in the example above. An example for condition #2 is {"John","Jerome","Smith"} and {"Mary"} in the following tree:



  In both cases, the proper noun corresponding to h_1 is the Subject, the proper noun corresponding to h_2 is the Object. In the case condition #1 holds, the Relation is defined to be h, while if condition #2 holds, the Relation is defined to be the concatenation of h and h'. If none of the condition holds, no triplet is outputted.

(c) **Evaluation:** in order to evaluate the extractors, apply the two extractors on the Wikipedia pages corresponding to:

- Donald Trump
- Brad Pitt
- Angelina Jolie

For each page, report the total number of triplets outputted by the extractors. In addition, for each of the two extractors take a random sample of 30 triplets, and manually verify whether the triplet is indeed valid or not.

For instance, ("Neil Gorsuch","appointed to","Court Supreme") is a valid triplet from the Wikipedia page of Donald Trump, as the page indeed mentions that Neil Gorsuch was appointed to the Supreme Court. The triplet ("Republican White House","combined with","Congress") is an invalid triplet because the text does not say that a Republican White House was combined with Congress or anything similar.[1]

(d) (Optional) Explain how to change the dependency-based extractor so that it could work in a head-final language, where the verb comes at the end of a clause (i.e., a Subject, Object, Verb order). What difficulty would a POS-based extractor might face in this case?

[1]It is a misanalysis of the sentence "Trump's victory marked the return of a Republican White House combined with control of both chambers of Congress".