Assignment Name: Dependency Parsing
Assignment number: 77

Constantin Mașca

Chapter 1

Final Report

1.1 Proposed problem

The problem addressed by this paper is information extraction from unstructured text. Specifically, given a sentence containing the recommended dose for a certain drug, the information to be extracted is the exact quantity expressed in milligrams/tablets/injections per day/hour/week etc.

1.2 Results

1.2.1 Explained Input

The input for the dependency parser is a set of roughly 1000 drug dosage descriptions. These were chosen from a larger set and all contain the keywords: "recommended dosage"/"recommended dose". The dependency parser then outputs a parse tree for each file in the input.

These parse trees are given as input to a tree distance calculator tool which was modified to accept a set of trees. The output of this tool is an $n \times n$ distance matrix containing the distance between each two of the n trees given as input.

The distance matrix is then fed into one of two custom clustering tools. Both of these are written in Python and use affinity propagation to group the trees based on their similarity.

The first clustering tool takes as input only one distance matrix and computes the clusters, while the second one takes as input a multitude of distance matrices with an increasing number of trees(50,100,150...1000).

1.2.2 Explained Output

A parse tree is a diagrammatic representation of the parsed structure of a sentence or string. Representing the dosage in this way is useful in order to be able to find similarities between them.

The matrix given as output by the tree distance software consists of two .csv files. The first one contains a list with the names of the drugs for which the distance was computed. The second file contains n columns and n rows. The values on row n represent the edit-distance from the nth tree to each of

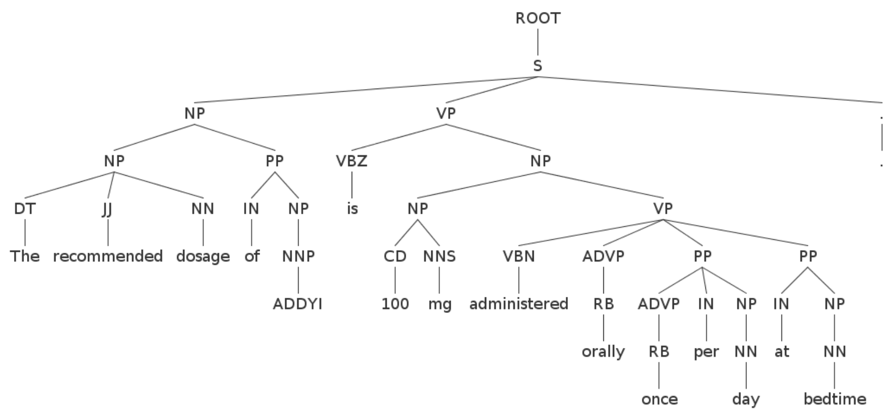

Figure 1.1: Parse tree structure for sentence: ”The recommended dosage of ADDYI is 100 mg administered orally once per day at bedtime.”

```
(ROOT
  (S
    (NP
      (NP (DT The) (VBN recommended) (NN dosage))
      (PP (IN of)
        (NP (NNP ADDYI))))
    (VP (VBZ is)
      (NP
        (NP (CD 100) (NN mg))
        (VP (VBN administered)
          (ADVP (RB orally))
          (PP
            (ADVP (RB once))
            (IN per)
            (NP
              (NP (NN day))
              (PP (IN at)
                (NP (NN bedtime))))))))
    (. .)))
```

Figure 1.2: Text format of the parse tree structure for the same sentence

the others. Edit-distance is defined as the total cost of transforming a source tree into a destination tree. Each operation of deleting, adding or renaming a node has a certain weight associated. Adding the cost for each operation used in a certain transformation gives the edit-distance between the two trees. The distance between a tree and itself is naturally 0.

The first clustering tool outputs the names of the trees in each cluster, with the exemplars (the most representative tree in the cluster) written in upper case, separating and numbering the clusters.

The second clustering tool was used for the visualization of the ratio of clusters / number of trees, outputting a plot of this relationship.

1.3 Results Analysis

The final result is the plot obtained by the second clustering tool (Figure 1.3):

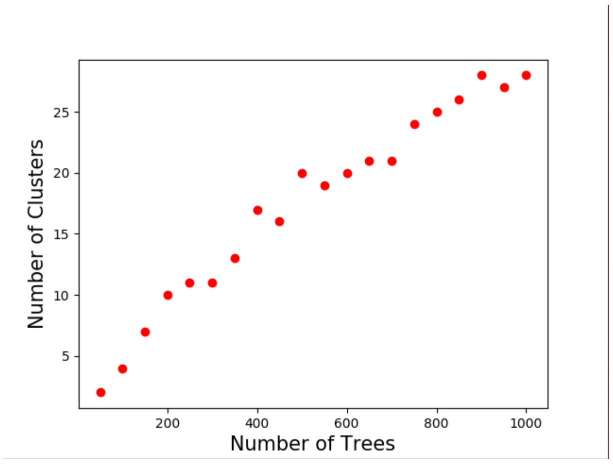

Figure 1.3: Relationship between the number of threes and the number of clusters

Generating the parse trees is relatively slow taking about an hour for all 1000 trees. One improvement that could be made is to modify the parser to accept multiple trees, eliminating the need for loading the application in memory for each file.

Calculating the distance between the trees is also slow, the algorithm used for the calculation of two trees being $\mathcal{O}(\mathrm{n}^2)$ (n being the number of nodes in the tree). Given the fact that n^2 calculations are needed for n trees, the average time taken for 1000 trees is about 15 minutes. This time could be cut in half by calculating only one side of this matrix (below or above the diagonal) and copying the data on the other side, the matrix being diagonally symmetric.

The actual clustering is quite fast, taking about 0.00065s on average for a number of 300 trees.

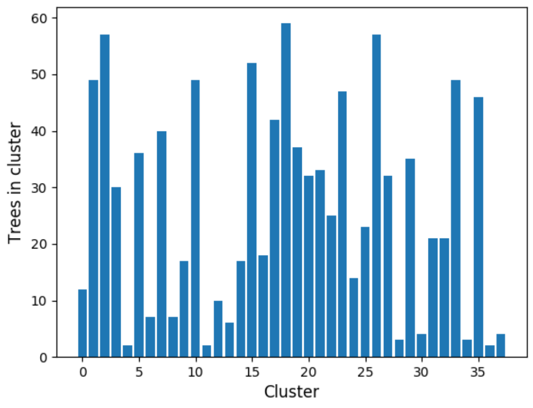

Figure 1.4: Histogram for the number of trees in each cluster

1.4 How to run the examples

Retrieving the data from the pharmaceutical website is done running the python written web-crawlers ”farm.py” (for retrieving all the pages containing drug descriptions) and ”spiderPig.py” (for retrieving the actual description from each page).

Generating the parse trees is done by running the ”convert.sh” bash script, which runs the dependency parser on the drug descriptions. ”convertTree.sh” is used for changing the format into an input that is accepted by the tree distance calculator.

Running the clustering tools requires Python 2.7+ installed on the machine. The command is

```
$python clusterer.py
```

for the first tool and

```
$python multipleClustering.py
```

for the second one.

1.5 Conclusions

Visualizing the graph, a logarithmic tendency can be easily observed, meaning that the problem proposed is probably realizable. Regardless, clustering a number of 1000 trees into 35 groups is a good starting point for information extraction in this domain. Rules can be made studying the exemplar of each cluster in order for any new sentence representing a drug dosage to be parsed.

Bibliography

https://nlp.stanford.edu/software/nndep.shtml

http://tree-edit-distance.dbresearch.uni-salzburg.at/

https://pdfs.semanticscholar.org/1ca8/d5b24b469b324a65e2968d2198226723974d.pdf