Lexicalized Parsing (LexPar)

Table of Contents

1 BASIC INFORMATION

1.1 Tool name

The tool is called **LexPar** which is the short for "Lexicalized Parsing" in Romanian and English.

1.2 Overview and purpose of the tool

LexPar performs a pseudo-syntatic analysis of a sentence using an improved version of the Lexical Attraction Models of Deniz Yuret (1998). It is a dependency-like analysis but without relation orientation and labeling. It is written in Perl and it is encapsulated as a SOAP web service which is WSDL described at http://ws.racai.ro/lxpws.wsdl. The version number of LexPar is currently 6.0.

1.3 A short description of the algorithm

Lexical Attraction Models (LAMs) were first introduced by Deniz Yuret (1998) to exemplify how an algorithm can learn word dependencies from raw text. The "syntactic structure" of a sentence is given by a dependency structure which is an undirected, connected and planar graph with no cycles. We will refer to this structure using the term "linkage" (not a proper syntactic structure as defined by Meľčuk (1988) because the links are not oriented and they have no names; what has been retained from Meľčuk's definition is the planarity condition which states that two links are not allowed to intersect except at a word position in a sentence). Yuret's general thesis is that lexical attraction is the likelihood of a syntactic relation. Therefore, the highest probability assigned by the model to a sentence is to be obtained by the syntactically correct linkage. He proves the result by formalizing the dependency structure of a sentence as a Markov network (or Markov random field (Kindermann and Snell 1950)). A node of the network is a word of the sentence and the potential function is the pointwise mutual information (MI) of a link.

LexPar (Ion and Barbu Mititelu, 2006) produces a linkage with the same properties as that of (Yuret 1998). It will be generated using the same basic idea: intermixing learning with parsing. The novelty of our approach resides in the use of with two additional devices:

1. linking rules that allow the formation of certain links while rejecting others;
2. use of POS information to increase the score of a weak (or unseen) link.

2 TECHNICAL INFORMATION

2.1 Software dependencies and system requirements

This kit contains a Perl wrapper to the LexPar linkage generator web service. The text to be analyzed is sentence split and then POS tagged and lemmatized. We (transparently) call the TTL

web service (also a deliverable of RACAI in Batch 2 of the MetaNet4U project) to achieve these annotations. To run these scripts one must install the following:

- `Perl`, a recent version from http://www.perl.org/get.html;
- Perl package `Unicode::String` from http://www.cpan.org/;
- Perl package `SOAP::Lite` from http://www.cpan.org/.

2.2 Installation

Other than what is mentioned in the previous section, there is no installation required for this tool.

2.3 Execution instructions

LexPar (script name '`lexparws.pl`') is called from a command line interface giving it two arguments: the language of the text to be linked, either English, ('`en`') or Romanain ('`ro`'), and the UTF-8 encoded text file to be processed. The distribution kit contains a test directory with two UTF-8 encoded text files: '`en-test-file.txt`' and '`ro-test-file.txt`'. If we want to run LexPar on the Romanian file, we should type:

```
./lexparws.pl ro test/ro-test-file.txt >result.txt
```

The output is written to the STDOUT in UTF-8 so that '`lexparws.pl`' can be integrated into command line pipelines. In the command above, the result is redicted into '`result.txt`' file in the same working directory as '`lexparws.pl`'.

2.4 Input/Output data formats

The input data to LexPar is a text file, UTF-8 encoded with no byte order markings. This file is pre-processed (internally) using the TTL web service.

The output data produced by LexPar is the text in the input file, sentence split, POS tagged and lemmatized, with linking information. Each token along with its annotations is printed on a line (separated by '`\r\n`'). Sentences are separated by a blank line (a single '`\r\n`').

Each token in a sentence is counted beginning with '`0`' up to the number of tokens in that sentence. The annotations of the current token are tab-separated ('`\t`') from the token and are (in the order of appearance): word form, POS tag, lemma, chunk and link index. The link index is the index of the word with which the current word links taking into acound the numbering of the tokens in the current sentence. **Please note that the link index may be missing** due to the fact that, in the process of linkage formation, that word could not be linked to any other word (e.g. some linking rule denied the formation of the link or the resulting graph would have not been planar). Also, the linkage annotation is not symmetric: if token A is linked to token B, then this information is present on either A or B, not both! See the next example for reference.

```
0     It              Pp3ns     it              Vp#1
1     helps           Vmip3s    help            Vp#1        0
2     the             Dd        the                         7
3     poor            Afp       poor            Ap#1        7
4     and             Cc-n      and                         5
5     many            Pi3-p     many            Np#1        7
6     middle-class    Afp       middle-class    Np#1,Ap#2   7
7     people          Ncn       people          Np#1        1
8     afford          Vmn       afford          Vp#2        1
9     the             Dd        the             Np#2        10
10    cost            Ncns      cost            Np#2        8
11    .               PERIOD    .
```

2.5 Integration with external tools

LexPar dependes on the availability of the TTL web service also hosted by RACAI to obtain sentence splitting, POS tagging and lemmatization. The WSDL of the TTL web service is located at http://ws.racai.ro/ttlws.wsdl.

3 CONTENT INFORMATION

3.1 Test input files

See the distribution kit in the '`test/`' directory. There is one test file for each of the languages supported by LexPar, namely English and Romanian.

3.2 Output files

For each input file in the '`test/`' directory, there is the corresponding output file in the '`sample-output/`' directory in the distribution kit.

3.3 Running times

The speeds that we are going to report have been obtained by calling the LexPar web service in the local network on a wireless connection of 54Mbps. One should consider the fact that the time the text is passed around the network is going to affect the overall performance of LexPar. Also, LexPar is running on a Ubuntu Server machine with a 8-core Intel(R) Xeon(R) CPU E5405 @ 2.00GHz CPU and 8 GB of RAM.

We have considered two types of measures: 'bytes/second' and 'tokens/second'. Bytes/second is going to help us predict the number of seconds after which an input UTF-8 text file of N bytes is going to be finished. Tokens/second gives the average processed tokens per second that the LexPar web service is capable. Please note that the LexPar web service also calls the TTL web sevice and its speed is going to be dependent on the TTL speed.

	English	Romanian
tokens/second	19	17
bytes/second	111	98

Table 1: LexPar speeds on each supported language

4 ADMINISTRATIVE INFORMATION

4.1 Contact

For further information, please contact Radu ION (radu@racai.ro).

5 REFERENCES

Ion, R., Barbu Mititelu, V. (2006). Constrained Lexical Attraction Models. In Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference. Menlo Park, Calif.: AAAI Press, pp. 297--302.

Kindermann, R. and Snell, J. L. 1950. Markov Random Fields and their Applications. American Mathematical Society, Providence Rhode Island.

Meľčuk, I. 1988. Dependency Syntax: Theory and Practice, New York , SUNY Press.

Yuret, D. 1998. Discovery of linguistic relations using lexical attraction. Ph.D. diss., Department of Computer Science and Electrical Engineering, MIT.