Linguakit

Developed by ProLNat@GE Group (http://gramatica.usc.es/pln/), CITIUS, University of Santiago de Compostela, Galiza.

LinguaKit is a Natural Language Processing tool containing several NLP modules (constantly updated and improved):

- Dependency parser (DepPattern)
- PoS tagger
- NER (named entity recognition)
- NEC (named entity classification)
- Coreference resolution of named entities
- Sentiment analysis
- Multiword extraction
- Keyword extraction
- Relation extraction
- Language recognition
- Tokenizer
- Sentence segmentation
- Lemmatization
- Keyword in context
- Entity linking and semantic annotation
- Summarizer
- Verb conjugator
- Language checker (spelling, lexicon, grammar)

Demo

A web interface to LinguaKit is available at LinguaKit.com

Description

The command linguakit is able to process 4 languages: Portuguese, English, Spanish and Galician. Since February 2018, a new language has been added: historical galician-portuguese (histgz), by Xavier Canosa, which is still a prototype that will be improved. The following tools are available. Scroll down for additional documentation and usage examples.

- **Dependency parser** (parameter dep): Runs parsers. The parsers are implemented in PERL and stored in the parsers file. The parsers were compiled from formal grammars (more information). There are several parameters to control output: basic triplets (-a), triplets with morphological information (-fa), the same output as the input (-c) for correction purpose, and CoNLL format (-conll). These parameters are further explained in the section Dependency Parser below.

- **PoS tagger** (parameter tagger): Provides the PoS tagger CitiusTools. It returns one PoS tag and one lemma per token. This is also known as PoS tagging disambiguation. The module is provided with two submodules: NER (-ner) and NEC (-nec). The NEC module returns semantic tags for named entities: NP00SP00 (Person), NP00G00 (Location), NP00O00 (Organization), NP00V00 (Miscellaneous).

- **COREF** (parameter coref) labels the different named entities of the text (identified by the NER and NEC) with a numeric id which represents the discourse entity they refer to (e.g., "Bob Marley NP00SP0 (1)", "Jimi Hendrix NP00SP0 (2)", "Marley NP00SP0 (1)", "Hendrix NP00SP0 (2)", etc.). COREF allows the -crnec option (experimental) which relabels some named entities based on the results of the coreference analysis. Please note that the COREF modules may slow down the execution of the system when analyzing large texts. Also, remember that COREF is performed document by document, so it is not recommended to run it in a large corpus containing several documents.

- **Multiword extraction** (parameter mwe): Extracts multiwords from PoS tagged text. There are several optional
parameters, each one being a specific lexical association measure for ranking the candidate terms: chi square (-chi, default), loglikelihood (-log), mutual information (-mi), symmetrical conditional probability (-scp), simple co-occurrences (-cooc).

- **Keyword extraction** (parameter key): Extracts keywords (lexemes and proper names) from PoS tagged text and ranked them using a reference corpus and chisquare.

- **Sentiment analysis** (parameter sent): Returns POSITIVE, NONE (neutral) or NEGATIVE, using a polarity lexicon and a classifier trained from annotated tweets. Given an input text, this module returns a polarity value for each paragraph, namely it returns three columns for each paragraph: the text in the first column, the polarity (pos, neg, none) in the second column, and the polarity score (from 0 to 1) in the third column. The last line of the output returns the overall score computed as the average of all paragraphs.

- **Relation extraction** (parameter rel): Returns triples SUBJECT - RELATION - OBJECT using methods based on Open Information Extraction.

- **Language recognition** (parameter recog): Returns the language of the input text: en, es, pt, gl, gz (agal galician variety), fr, eu, ca, bn (bengali), ur (urdu), hi (hindi), ta (tamil). This module is also used by other modules to recognize the language before processing (only for the supported languages: pt, en, es, gl).

- **Tokenizer** (parameter tok): Returns a tokenized text. Parameter -split splits word contractions and verb clitics. Parameter -sort ranks tokens by frequency.

- **Sentence segmentation** (parameter seg): Returns a sentence per line. Sentence segmentation is the problem of dividing a string of written language into its component sentences.

- **Lemmatization** (parameter lem): Returns all the lemmas of each token and all the morphological information (PoS tags) associated to each lemma. This is the process running before PoS tagging disambiguation.

- **Keyword in context** (parameter kwic): Returns a target word in context (window: 10 tokens). Option -tokens returns tokens as context. This module requires the keyword to be searched as an additional argument.

- **Entity linking** (parameter link): Returns a list of terms which represent Wikipedia entities. Besides, the input text is annotated with those terms and their links to Wikipedia. Requires Internet connection since it runs via Web API service. The output can be in two formats: json (default) and xml.

- **Summarizer** (parameter sum): Returns an abstract of the input text. You can choose the percentage of the text to be summarized by using as option a number from 1 to 100. The code was developed by Fernando Blanco Dosil when it was working in Cilenis Language Technology.

- **Conjugator** (parameter conj): Returns the verb inflection if you enter the infinitive form. Pay attention that the input is not a file but a string, the infinitive verb, and the module should be used like this: ./linguakit conj pt "fazer" -s -pb. The module is working for three languages: Galician, Spanish and Portuguese. In the case of Portuguese verbs, you can choose among 4 language varieties: european portuguese after the spelling agreement (-pe), brasian portuguese after the spelling agreement (-pb), european portuguese before the spelling agreement (-pen), brasian portuguese before the spelling agreement (-pbn). The output is in json format. This module requires Internet connection since it runs using a Web API.

- **Language checker** (parameter aval): Returns the language errors found in the input sentence. Several types of errors are considered: spelling, lexical, and grammatical issues. Suggestions of correction are provided as well as a linguistic explanation for each specific type of error. Requires Internet connection since it runs via Web API service. The output can be in two formats: json (default) and xml. By now, it is only available for Galician language (gl).

**Requirements**

Depending on your GNU/Linux version or distribution, you may need to install some CPAN Perl modules: *Getopt::ArgParse* Perl module. *LWP::UserAgent* Perl module. *HTTP::Request::Common* Perl module. *Storable* Perl module.

To install them, you may use the -MCPAN interface at the command line:

```sh
sudo perl -MCPAN -e 'install (Getopt::ArgParse)'
sudo perl -MCPAN -e 'install (LWP::UserAgent)'
sudo perl -MCPAN -e 'install (HTTP::Request::Common)'
sudo perl -MCPAN -e 'install (Storable)'
```
Installation

Using Git

```bash
git clone https://github.com/citiususc/Linguakit
```

ZIP Download

Download Linguakit-master.zip and then:

```bash
unzip Linguakit-master.zip
```

Using Make (to be installed in an accessible bin directory):

```bash
git clone https://github.com/citiususc/Linguakit
cd Linguakit
sudo make deps
sudo make install
sudo make test-me
```

Thanks to José João Almeida (Univ. do Minho) for the Make file.

As all modules are been updated regularly, you’d better use `git` to install and update the system.

A New more efficient version of LinguaKit was released in 22 June 2017 by César Piñeiro (also for Windows `linguakit.bat` command).

Usage (Linux)

Run `.linguakit --help` to see the options of a module. These are the available command-line options:

```bash
./linguakit <module> <lang> <input> [options]
cat <input> | ./linguakit <module> <lang> [options]
```

- `module`: dep, tagger, mwe, recog, sent, rel, tok, seg, kwic, link, sum, conj
- `language`: gl, es, en, pt, histgz
- `input`: path of the input (by default a txt file or gz/zip)

- `dep`: dependency syntactic analysis
- `tagger`: part-of-speech tagging
- `mwe`: multiword extraction
- `key`: keyword extraction
- `recog`: language recognition
- `sent`: sentiment analysis
- `rel`: relation extraction
- `tok`: tokenizer
- `seg`: sentence segmentation
- `lem`: lemmatization
- `kwic`: keyword in context (concordances)
- `link`: entity linking and semantic annotation
- `sum`: text summarizer
- `conj`: verb conjugator (the input is just a verb)
- `coref`: named entity coreference solver
- `aval`: language checker: avalingua

Run `.linguakit <module> --help` to see the options of a module. These are the available command-line options:
Usage in Windows

The same syntax with linguakit.bat command. You must install Perl and insert the path for the interpreter.

Examples

Return a dependency-based analysis in CoNLL format:

```
./linguakit dep pt test/pt.txt -conll
```

Return the PoS tags with NEC information for named entities (in historical galician-portuguese - histg2):

```
./linguakit tagger histg2 test/histg2.txt -nec
```

Return the PoS tags with NEC information for named entities and Coreference Resolution:

```
./linguakit coref en test/en.txt
```

Return a sentiment value:

```
./linguakit sent en "I don't like the film" -s
echo  "I don't like the film" | ./linguakit sent
```
Make multiword extraction ranked with chi-square:

```
./linguakit mwe pt test/pt.txt -chi
```

Generate the context in tokens of the keyword *presidente* (concordances or keyword in context).

```
./linguakit kwic pt test/pt.txt -tokens "presidente"
```

Return triples (relations):

```
./linguakit rel en test/en.txt
```

Return an abstract or summary of the input text (50%):

```
./linguakit sum en test/en.txt -p 50
```

Return the european portuguese inflection of the input verb:

```
./linguakit conj pt "fazer" -s -pe
echo "fazer" | ./linguakit pt conj -pe
```

**Input file**

The input must be in plain text format, and encoded in UTF8.

**Lexicons**

Lexicons (electronic dictionaries) are in `tagger/*/lexicon/dicc.src` files. If you modify them, then you should recompile them by running:

```
./lexicon_compiler.sh
```

**Dependency parser**

**Output format**

Parameter `-a` gives as output the basic dependency-based analysis. Each analysed sentence consists of two elements:

1. A line containing the POS tagged lemmas of the sentence. This line begins with the tag `SENT`. The set of tags used here are listed in file `TagSet.txt`. All lemmas are identified by means of a position number from 1 to N, where N is the size of the sentence.
2. All dependency triplets identified by the grammar. A triplet consists of: 

   `(relation;head_lemma;dependent_lemma)`

For instance, the sentence *"I am a man."* generates the following output using the parameter `-a`:

```
SENT::<I_PRO_0_<number:0|lemma:I|possessor:0|case:0|genre:0|person:0|politeness:0|type:P|token:I|>
am_VERB_1_<number:0|mode:0|lemma:be|genre:0|tense:0|person:0|type:S|token:am|>
a_DT_2_<number:0|lemma:a|possessor:0|genre:0|person:0|type:0|token:a|>
man_NOUN_3_<number:S|lemma:man|genre:0|person:3|type:C|token:man|> _SENT>
(Lobj:be_VERBF_1_i_PN_0)
(Spec:man_NOM_3:a_DT_2)
(Robj:be_VERBF_1:man_NOM_3)
```

Parameter `-fa` gives rise to a full representation of the dependency-based analysis. Each triplet is associated with two pieces of information: morpho-syntactic features of both the head and the dependent.
Parameter -c generates as output with the same format as the input (a tagged text) but with some corrections proposed by the grammar. This option is useful to identify and correct regular errors of PoS taggers using grammatical rules.

Parameter -conll generates an output with the format defined by CoNLL-X, inspired by Lin (1998). This format was adopted by the evaluation tasks defined in CoNLL.

**PoS tagger**

The EAGLES convention is being followed.

**Coreference Resolution**

More information »

**Sentiment analysis**

The input can be either a file (by default) or a string (option -e). The output is POSITIVE, NONE, OR NEGATIVE, and a score between 0 and 1. The classifier was trained with tweets, so the input should be just one sentence or a small paragraph.

More Information »

**Multiword Extraction**

More Information »

**Language identification**

More Information »

**Entity linking**

This module reads the input text and returns three types of terms and concepts:

- **MAIN_TERM** is a term occurring in the text that is linked to a Wikipedia concept.
- **NEW_TERM** is a Wikipedia concept which does not occur in the text but is semantically related to some of the main terms.
- **CATEGORY** is a Wikipedia category used to categorize and classify the text.

Also returns an annotated version of the input text, namely the text is annotated with tags identifying the main terms. Each MAIN_TERM is enriched with a link to its corresponding concept in Wikipedia.

For more information, you can look up for our paper:


**Additional Documentation and Bibliography**

More information on the modules can be found in papers stored in the docs directory.

**References**

The toolkit


Dependency analysis


PoS tagging, NEC and Coreference Resolution


Sentiment analysis


Multiword extraction


Relation extraction


Entity linking

Language checker (Avalingua)


INFORMACIÓN

Investigadores
Pablo Gamallo Otero

Licença

DESCARGAR

🔗 Repositorio Gitlab
⬇️ Descargar de Gitlab
🔗 Repositorio Github

PUBLICACIÓN

Multilingual Open Information Extraction

PROXECTOS DE INVESTIGACIÓN

TELEPARES: Tecnologías de la lengua para análisis de opiniones en redes sociales

DEMOSTRADORES

Linguakit