TaCo – readme

June 29, 2012

Bartosz Zaborowski

Instytut Podstaw Informatyki Polskiej Akademii Nauk

About

TaCo is a statistical morphosyntactic tagset converter designed for positional tagsets, especially Polish tagsets. It's typical use is to convert manual annotation of a corpus with tags from one morphosyntactic tagset to another. It is based on decision trees produced by C5.0 algorithm and additionally makes use of morphological analyzer Morfeusz. TaCo has been developed at the Institute of Computer Science, Polish Academy of Sciences, Warsaw.

TaCo official homepage: http://zil.ipipan.waw.pl/TaCo/

Author: Bartosz Zaborowski [bartosz.zaborowski@ipipan.waw.pl]

License: GPL v.3

1 Prerequisites

- Ruby interpreter, version 1.9.1 or newer
- C5.0, Release 2.07 GPL Edition (other versions may work if the output of the tool do not change). **C5.0 is only needed to train a new conversion model. It is not needed for converting tagset using already trained model.**
- Morfeusz SGJP morphological analyzer, version 0.82 <2010/02/22>(other versions may work if the input/output format of the tool do not change)

** Although TaCo does not depend on particular operating system, it is probably hard to use under non-UNIX systems (e.g. MS Windows) because of need of compiling the C5.0. **

2 Standards for representation of linguistic information

2.1 Tagsets

Tagsets used by converter are defined in a format similar to the format used by Spejd[1]. TaCo's package contains 2 examples of such tagset definitions, one of them (*kipi2nkjp/nkjp.cfg* is documented in comments).

2.2 Input and output formats

TaCo can read and write XCES corpus xml format (as in IPIPAN [2] Corpus, or as used by Spejd). The package contains some examples of such files.

[1] http://zil.ipipan.waw.pl/Spejd

[2] IPIPAN - Institute of Computer Science, Polish Academy of Sciences

3 Installation

3.1 Prerequisites

Linux The ruby interpreter should be available through the package manager in most Linux distributions. The Morfeusz binary for Linux can be downloaded from http://sgjp.pl/morfeusz/. The C5.0 has to be compiled by hand. Assuming that there is a make and gcc available in the system, it is fairly simple. Download it from C5.0 homepage (http://rulequest.com/download.html) or TaCo page, then unpack the C5.0 package and run make like this:

```
tar xzf C50.tgz
make
```

You can install the c5.0 system-wide by copying the c5.0 file to e.g. /usr/local/bin or any other directory from *$PATH* (requires superuser privileges).

Windows Probably the simplest way to setup the C5.0 is to use MinGW (http://www.mingw.org/wiki/Getting_Started) and compile the C5.0 sources in the MinGW shell. Download the C5.0 from its homepage (http://rulequest.com/download.html) or TaCo page. Then unpack and type *make*, just like under Linux.

Ruby interpreter can be downloaded from http://www.ruby-lang.org/. The Morfeusz binary for windows is available at it's homepage (http://sgjp.pl/morfeusz/).

3.2 TaCo

TaCo is ready for use after extracting the package. Just remember to set correct paths to the c5.0 and morfeusz binaries in the configuration file.

4 Basic usage

TaCo is a command line tool. It doesn't have any graphical interface.

The basic usage help is displayed when TaCo is run without arguments.

If you only want to convert a corpus from one tagset to another with an already trained model, read the section 4.1. If you want to train a conversion model on a new pair of tagsets, read section 4.2.

For Windows users: you have to start each of the following commands by path to ruby interpreter (*ruby.exe* will be enough in most cases).

4.1 Conversion/applying the model

The command converting annotation of a corpus to the other tagset looks like this:

```
path/to/taco path/to/configuration/file path/to/model path/to/source/corpus \
source_corpus_filename path/to/source/tagset/definition target_corpus_filename \
path/to/target/tagset/definition
```

Lets assume we have a model named *kipi2nkjp.tc*, configuration file *kipi2nkjp.conf*, the source corpus located in *corpus/* directory containing *morph.xml* files and definitions of tagsets named *kipi.cfg* and *nkjp.cfg*. Then, the command which will write annotations along the source corpus in *morphOut.xml* files will be:

```
../taco kipi2nkjp.conf kipi2nkjp.tc corpus/ morph.xml kipi.cfg morphOut.xml nkjp.cfg
```

The above command can be executed on the example included in the TaCo package. (possibly after correcting path to morfeusz in kipi2nkjp.conf). It is located in the *kipi2nkjp/* directory.

4.2 Training the model

The command for training a model on a pair of corpora annotated by means of two tagsets is:

```
path/to/taco -t path/to/configuration/file path/to/model path/to/source/corpus \
source_corpus_filename path/to/source/tagset/definition path/to/target/corpus \
target_corpus_filename path/to/target/tagset/definition
```

Lets assume we have a configuration file *kipi2nkjp.conf*, the source corpus located in *corpus2/kipi* directory in *morph_kipi.xml* files, the target corpus located in *corpus2/nkjp* directory in *morph_nkjp.xml* files and definitions of tagsets named *kipi.cfg* and *nkjp.cfg*. Then, the command which will train a model of conversion and save it to *kipi2nkjp_new.tc* file will be:

```
../taco -t kipi2nkjp.conf kipi2nkjp_new.tc corpus2/kipi morph_kipi.xml kipi.cfg \
corpus2/nkjp morph_nkjp.xml nkjp.cfg
```

This command will work in the example directory.

4.3 Evaluation and parameter tuning

The configuration file allows to change values of multiple parameters of training. It may be useful for tuning TaCo for the best performance on different pairs of tagsets. To simplify evaluation of particular parameter configuration you can use an *evaluation* tool included in the TaCo package. The invocation is similar to training mode of TaCo, it only does not contain model name and -t parameter:

```
path/to/evaluation path/to/configuration/file path/to/source/corpus \
source_corpus_filename path/to/source/tagset/definition path/to/target/corpus \
target_corpus_filename path/to/target/tagset/definition
```

The *evaluation* tool performs a crossvalidation using annotation from source and target corpora as gold standard.

For the example from the previous section it may be:

```
../evaluation kipi2nkjp.conf corpus2/kipi morph_kipi.xml kipi.cfg \
corpus2/nkjp morph_nkjp.xml nkjp.cfg
```

4.4 Model inspecting

An another simple tool allows to display in ASCII-art the decision tree(s) used in the conversion model. The command for it is:

```
path/to/print_tree path/to/model max_depth_to_print
```

The tree is displayed vertically (root at the left, leafs at the right).

For the examples from the *kipi2nkjp* directory it can be:

```
../print_tree kipi2nkjp.tc 3
```

5 Additional documentation

Formats of the configuration file and tagset definitions are documented in the example configuration (*kipi2nkjp/kipi2nkjp.conf*) and one of the example tagset definitions (*kipi2nkjp/nkjp.cfg*).

6 Examples

The example directory *kipi2nkjp* contains a configuration, tagset definitions, tiny fragments of the Frequency Corpus of Polish and a trained model for conversion from IPIPAN Corpus tagset to National Corpus of Polish tagset.