Short Manual for Mate Tools

Wolfgang Seeker

September 9, 2013

Contents

Warning

This manual may change if the tools are updated.

1 Preliminaries

Mate tools is implemented in Java and runs on any machine that has a Java Runtime Environment. In order to use mate tools, download the jar-file from `code.google.com/p/mate-tools`.

At the writing of this manual, the most recent version is **anna-3.3.jar**.

Mate tools provides four different processing tools: a **lemmatizer**, a **part-of-speech tagger**, a **morphology tagger**, and a **dependency parser**. All tools are purely data-driven and do not need any additional resources to run. In the full chain, they are applied in the given order, each reading the output of the previous tools.

Note that there is **no tokenizer included**, which means that you need to provide already tokenized input for the tools.[1]

All tools use the CoNLL 2009 Shared Task format, which consists of at least 12 tab-separated columns for each token. Each token of a sentence is represented

[1]For the best results, the tokenization needs to be the same as the tokenization in the treebank that your models were trained on.

by one line, sentences are separated by an empty line. The columns specify the following information in that order: word id, word form, gold lemma, predicted lemma, gold POS, predicted POS, gold features, predicted features, gold head, predicted head, gold label, predicted label. You can add additional columns to the end, which will simply be copied by the tools. In each column except the first, you can give an underscore to mark empty fields. The feature columns can be lists of arbitrary features separated by the pipe character (e.g. *nom*|*sg*|*masc*).

The tools will always write to the predicted column of their associated information (i.e. the lemmatizer writes to predicted lemmas, the POS tagger writes to predicted POS). Tools later in the chain will read the predicted columns from earlier tools (i.e. the parser will use information from predicted lemmas, predicted POS, and predicted features).

2 Running Mate Tools from Command Line

For running the tools, you first need a trained model. Models for several languages are provided on the mate tools website `code.google.com/p/mate`.

Run the tools with the following commands:

- Lemmatizer:
  ```
  $>java -cp anna-3.3.jar is2.lemmatizer.Lemmatizer -model <modelfile>
  -test <inputfile> -out <outputfile>
  ```
- POS tagger:
  ```
  $>java -cp anna-3.3.jar is2.tag.Tagger -model <modelfile> -test <inputfile>
  -out <outputfile>
  ```
- Morphology tagger:
  ```
  $>java -cp anna-3.3.jar is2.mtag.Tagger -model <modelfile> -test <inputfile>
  -out <outputfile>
  ```
- Dependency parser:
  ```
  $>java -cp anna-3.3.jar is2.parser.Parser -model <modelfile> -test <inputfile>
  -out <outputfile>
  ```

For example, if you want to run the part-of-speech tagger on a file named **dev.conll09.lemmatized** using a model file called **postagger.model** and write the output to a file named **dev.conll09.lemmatized.postagged**, you would use the following call:

```
$>java -cp anna-3.3.jar is2.tag.Tagger -model postagger.model
-test dev.conll09.lemmatized -out dev.conll09.lemmatized.postagged
```

There is an additional switch for the lemmatizer (**-uc**) that switches on upper-case handling. This was added to handle the capitalization of nouns in German and makes sure that the lemmata of these words will also start with an upper-case character. If capitalization is important to your lemmatization task, add this switch to the call above.

3 Training Your Own Models

For training your own models, you first need a treebank in CoNLL 2009 Shared Task format. The tools will use their corresponding gold columns as labels for their training examples. For features provided by previous tools in the chain, they will use the predicted columns. For best results, perform jackknifing on your training data to provide realistic features for the tools later in the chain.

Train your own models with the following commands:

- Lemmatizer:
  $>java -cp anna-3.3.jar is2.lemmatizer.Lemmatizer -train <trainingdata> -model <modelfile>
- POS tagger:
  $>java -cp anna-3.3.jar is2.tag.Tagger -train <trainingdata> -model <modelfile>
- Morphology tagger:
  $>java -cp anna-3.3.jar is2.mtag.Tagger -train <trainingdata> -model <modelfile>
- Dependency parser:
  $>java -cp anna-3.3.jar is2.parser.Parser -train <trainingdata> -model <modelfile>