# Tools for automatic annotation of Swedish and Turkish texts

Eva Pettersson
evapet@stp.lingfil.uu.se
Uppsala University

November 29, 2007

# Contents



# List of Figures



# 1 Introduction

The project *Supporting Research Environment for Minor Languages* aims at building parallel corpora for various less explored languages, to promote research and teaching for these languages. As a pilot study within the project, a parallel Swedish-Turkish treebank is developed using existing tools for creating the treebank and for the visualisation of the results.

For the creation of the Swedish-Turkish treebank, tools are needed for Swedish and Turkish part-of-speech tagging and syntactic annotation as well as for automatic sentence and word alignment. For the manual correction, interactive tools are needed for visualising and correcting part-of-speech tagging and syntactic annotation as well as sentence and word alignment.

To facilitate the automatic annotation and alignment process, a graphical interface, Uplug Connector, was developed by Bengt Dahlqvist [5]. At present, the interface is adapted to Swedish and Turkish, but the modularity of the interface makes it fairly easy to adapt the interface to other languages as well.

In this report I will describe the scripts that I developed for automatic annotation and parsing of Swedish and Turkish texts, using existing tools. The scripts are an integrated part of the Uplug Connector, but may also be run as separate modules.

# 2 The annotation process

All texts in the parallel treebank are automatically annotated with part-of-speech and morpho-syntactic information as well as syntactic dependency rela-

tions.

The annotation process described in this report presumes that the input texts are formatted using the XML Corpus Encoding Standard (XCES). The appropriate XCES format may be achieved from plain texts by running the preprocessing module integrated in the Uplug Connector[5]. An example XCES representation of a Swedish sentence in one of the treebank texts is illustrated below:

```
<p id="2">
<s id="s2.1">
<w id="w2.1.1">Till</w>
<w id="w2.1.2">fängelset</w>
<w id="w2.1.3">förde</w>
<w id="w2.1.4">de</w>
<w id="w2.1.5">bara</w>
<w id="w2.1.6">ett</w>
<w id="w2.1.7">fåtal</w>
<w id="w2.1.8">fångar</w>
<w id="w2.1.9">.</w>
</s>
</p>
```

The annotation script will produce an output file identical to the input file, but with an additional `pos` feature for each word, containing the resulting part-of-speech tag proposed by the tagger:

```
<p id="2">
<s id="s2.1">
<w pos="PP" id="w2.1.1">Till</w>
<w pos="NN_NEU_SIN_DEF_NOM" id="w2.1.2">fängelset</w>
<w pos="VB_PRT_AKT" id="w2.1.3">förde</w>
<w pos="PN_UTR@NEU_PLU_DEF_SUB" id="w2.1.4">de</w>
<w pos="AB" id="w2.1.5">bara</w>
<w pos="DT_NEU_SIN_IND" id="w2.1.6">ett</w>
<w pos="NN_NEU_SIN_IND_NOM" id="w2.1.7">fåtal</w>
<w pos="NN_UTR_PLU_IND_NOM" id="w2.1.8">fångar</w>
<w pos="MAD" id="w2.1.9">.</w>
</s>
</p>
```

The script also produces an output file with syntactic dependency relations along the part-of-speech tags. The dependency relations are represented as a `head` feature, stating the word number that the current word is modifying, and a `deprel` feature, stating the dependency relation between the current word and its head word:

```
<p id="2">
<s id="s2.1">
<w pos="PP" head="3" deprel="ADV" id="w2.1.1">Till</w>
<w pos="NN_NEU_SIN_DEF_NOM" head="1" deprel="PR" id="w2.1.2">fängelset</w>
<w pos="VB_PRT_AKT" head="0" deprel="ROOT" id="w2.1.3">förde</w>
```

```
<w pos="PN_UTR@NEU_PLU_DEF_SUB" head="3" deprel="SUB" id="w2.1.4">de</w>
<w pos="AB" head="3" deprel="ADV" id="w2.1.5">bara</w>
<w pos="DT_NEU_SIN_IND" head="7" deprel="DET" id="w2.1.6">ett</w>
<w pos="NN_NEU_SIN_IND_NOM" head="8" deprel="DET" id="w2.1.7">fåtal</w>
<w pos="NN_UTR_PLU_IND_NOM" head="3" deprel="OBJ" id="w2.1.8">fångar</w>
<w pos="MAD" head="3" deprel="IP" id="w2.1.9">.</w>
</s>
</p>
```

An equivalent Tiger XML version of the dependency parsing is also created by the script. This file is useful for visualising the annotation results using existing tools such as TIGERSearch[3].

When running the annotation scripts, you need to make sure that there is a file `Uplugconn.cfg` in the directory where you run the program. This file is the configuration file, with paths to external programs used in the annotation process. Since the configuration file is the same as the one used in Uplug Connector, some lines are not used specifically by the annotation scripts. Whether you run the annotation scripts as separate modules or as an integrated part of Uplug Connector, the configuration file should contain six lines, in the following order:

1. Path to the directory where Clue Aligner is installed. Default value is: /local/ling/uplug/
2. Path to the Clue Aligner program. Default value is: /local/ling/uplug/uplug
3. Path to tools used by Uplug Connector. Default value is: /local/ling/parcor/tools/uplugconnector/
4. Path to the working directory for the annotation of Swedish texts. The Swedish annotation script is expected to be in this directory, as well as several other programs that the annotation script is depending on. Default value is: /local/ling/parcor/tools/sweparse
5. Path to the working directory for the annotation of Turkish texts. The Turkish annotation script is expected to be in this directory, as well as several other programs that the annotation script is depending on. Default value is: /local/ling/parcor/tools/turparse
6. Path to the directory where the tagger is installed. Default value is: /local/ling/tnt/tnt/

## 2.1 Annotation of Swedish texts

The Swedish annotation script is a shell script (`sweparse.sh`), that is expected to be in the working directory specified on the forth line in the configuration file (`Uplugconn.cfg`).

For the morpho-syntactic annotation, the script uses the TnT tagger[**?**] trained for Swedish on the SUC corpus, with an accuracy of 96%[4].

The syntactic annotation is performed using MaltParser 0.4 with a pre-trained SVM parsing model for Swedish[6].

The script requires five arguments, in the below specified order:

1. input file in above specified XCES format
2. output file for the part-of-speech annotation (XCES format)
3. output file for the syntactic annotation (XCES format)
4. input encoding
5. output encoding

A third output file is also created, with the syntactic annotation in Tiger XML-format. This file will receive the same name as the syntacically annotated XCES file, except for the additional ending *_tiger*.
In chapter 2.1.1 I will describe the different steps of the script in more detail.

### 2.1.1 Steps in the Swedish annotation process

The Swedish annotation script involves the following steps:

1. The working directory where the Swedish annotation script and related programs are located is set to the directory specified on the fourth line in the configuration file (`Uplugconn.cfg`).
2. The directory for running the tagger is set to the directory specified on the last line in the configuration file.
3. The text is recoded from its original encoding to ISO-8859-1, to conform to the tagger.
4. The text is normalised to the format appropriate for the TnT tagger, by removing the XML-tags, leaving one word per line.
5. The text is tagged using the TnT tagger.
6. A Perl program, `putTagsInXml.pl`, is called to insert the resulting tags into the original XCES-file. The part-of-speech tags are inserted in the `pos` feature for each word.
7. The resulting XCES-file is recoded to the desired output format specified by the user when starting the program.
8. A Perl program, `convert_tagformat.pl`, is called to convert the tagged XCES-file to Malt-TAB format, as to prepare for the syntactic parsing.
9. MaltParser presumes that the part-of-speech tags conform to the Granska tag set. The SUC tags are converted to Granska tags in two steps. First a mapping program developed by Johan Hall (Växjö University) is used. Since there are a few cases not covered by this program, an additional perl program was created for the cases not covered by the mapping program: `postconvert_taggformat.pl`.

10. The MALT parser is run in two modes, creating one output file in MaltTAB format and one output file in Tiger XML.

11. A Perl program, `putDepsInXml.pl`, is called to insert the dependency relations into the tagged XCES-file. The dependencies are inserted in the `head` and `deprel` features for each word, where *head* specifies the word that the current word is modifying and *deprel* specifies the dependency relation between the current word and its head word.

12. Both the resulting XCES file and the Tiger XML version are recoded to the desired output format specified by the user when starting the program.

13. All temporary files that were created during the annotation process are removed, so that there are three output files remaing: the part-of-speech tagged XCES-file, the syntacically parsed XCES-file and the Tiger XML version of the syntactically parsed text.

Making use of the Tiger XML output file, the result of the morphological and syntactic parsing may be easily visualised using for example TIGERSearch[3], as illustrated in figure 1.

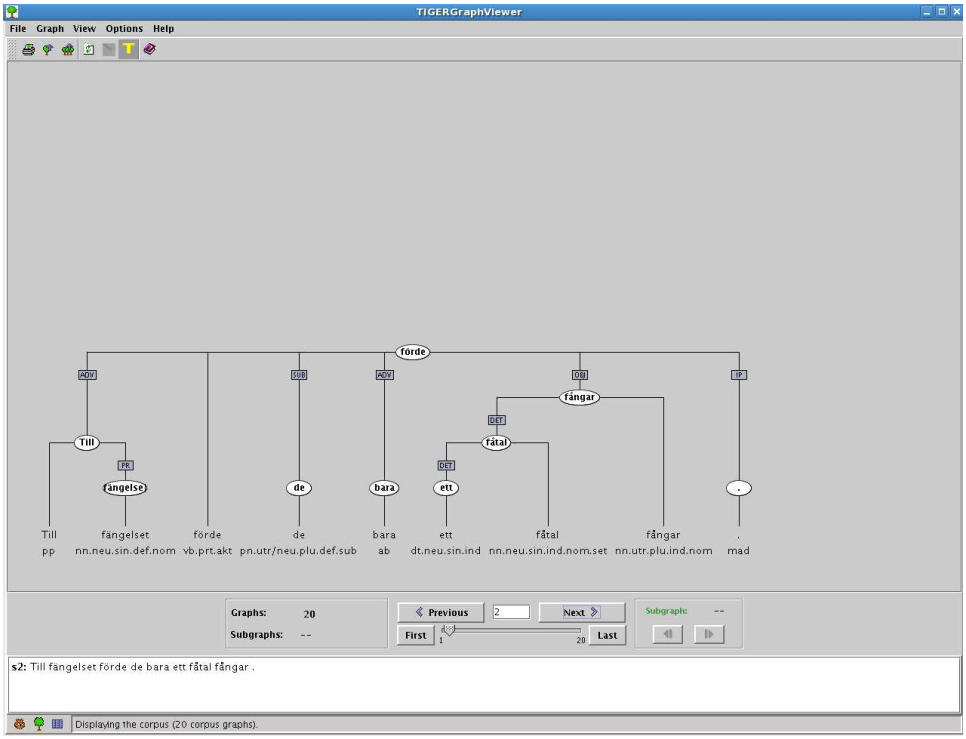


Figure 1: A morphologically and syntactically annotated Swedish sentence visualised in TIGERSearch

## 2.2 Annotation of Turkish texts

The Turkish annotation script is a shell script (`turparse.sh`), that is expected to be in the working directory specified on the fifth line in the configuration file (`Uplugconn.cfg`).

For the morpho-syntactic annotation, the script uses an automatic morphological segmenter and analyser developed for Turkish by Kemal Oflazer, with an accuracy of 74% leaving ambiguous tokens.[8] The disambiguation is then performed using the disambiguator described by Yüret and Türe. The disambiguator automatically learns morphological disambiguation rules from a decision list induction algorithm, with an average accuracy of 96%.[10]

The syntactic annotation is performed using MaltParser 0.4 with a pretrained parsing model for Turkish[2][7].

The script requires five arguments, in the below specified order:

1. input file in above specified XCES format
2. output file for the part-of-speech annotation (XCES format)
3. output file for the syntactic annotation (XCES format)
4. input encoding
5. output encoding

A third output file is also created, with the syntactic annotation in Tiger XML-format. This file will receive the same name as the syntacically annotated XCES file, except for the additional ending *_tiger*.
In chapter 2.2.1 I will describe the different steps of the script in more detail.

### 2.2.1 Steps in the Turkish annotation process

The Turkish annotation script involves the following steps:

1. The working directory where the Turkish annotation script and related programs are located is set to the directory specified on the fifth line in the configuration file (Uplugconn.cfg).
2. The text is recoded from its original encoding to ISO-8859-9, to conform to the tagger.
3. The text is normalised to the format appropriate for the morphological analysis and disambiguation, by the perl program `reformat.pl`. The normalisation includes removal of XML-tags from the XCES-file leaving one word per line, removal of empty lines, removal of sequential whitespace characters and insertion of the sentence delimiter *******.
4. The normalised text is morphologically analysed and disambiguated using the analyser developed by Kemal Oflazer[8] and the disambiguator described in [10]. The analysis and disambiguation process is performed by a Perl script, `prepareconllfromsentence.pl`, originally developed by Gülşen Eryiğit (Istanbul Technical University, Department of Computer Engineering) and slightly adapted to our annotation process by me.

5. A Perl program, `putTagsInXml.pl`, is called to insert the resulting tags into the original XCES-file. The part-of-speech tags are inserted in the `pos` feature for each word.

6. The resulting XCES-file is recoded to the desired output format specified by the user when starting the program.

7. The tagged file is recoded to UTF-8, to conform to the parser.

8. The MALT parser is run in two modes, creating one output file in Malt-TAB format and one output file in Tiger XML.

9. The Tiger XML file produced by the MALT parser only contains the actual part-of-speech, omitting all morphological features. The script `expand_postags.pl` was therefore developed to include also the morphological features in the Tiger XML file.

10. A Perl program, `putDepsInXml.pl`, is called to insert the dependency relations into the tagged XCES-file. The dependencies are inserted in the `head` and `deprel` features for each word, where *head* specifies the word that the current word is modifying and *deprel* specifies the dependency relation between the current word and its head word.

11. Both the resulting XCES file and the Tiger XML version are recoded to the desired output format specified by the user when starting the program.

12. All temporary files that were created during the annotation process are removed, so that there are three output files remaing: the part-of-speech tagged XCES-file, the syntacically parsed XCES-file and the Tiger XML version of the syntactically parsed text.

Making use of the Tiger XML output file, the result of the morphological and syntactic parsing may be easily visualised using for example TIGERSearch[3], in the same way as illustrated for Swedish in figure 1.

# 3 Where to find the scripts

The annotation scripts and the programs used by the scripts are available from the local cvs directory at the Department of Linguistics and Philology at Uppsala University:

- Directory with the Swedish annotation script and related programs: /local/ling/parcor/tools/sweparse/
- Directory with the Turkish annotation script and related programs: /local/ling/parcor/tools/turparse/

The annotation scripts may also be run as an integrated part of Uplug Connector, as descibed in chapter 4.

# 4 Integration in Uplug Connector

The scripts for annotating Swedish and Turkish texts morphologically and syntactically may be run as separate terminal-based scripts or as an integrated part of the graphical interface Uplug Connector[5].

In Uplug Connector, the Swedish annotation script is called by selecting the option *SweDepParse MALT*. For Turkish annotation the option *TurkDepParse MALT* should be chosen. Input and output files are selected with the leftmost buttons in the Uplug Connector interface, whereas input and output encoding is chosen from selection menus to the right of these buttons (see figure 2).

Uplug Connector also offers features such as preprocessing and sentence and word alignment, using Jörg Tiedemann's Clue Aligner[9]. See further [5].

Uplug Connector is available from the local cvs directory at the Department of Linguistics and Philology at Uppsala University:
/local/ling/parcor/tools/uplugconnector/upc112.jar

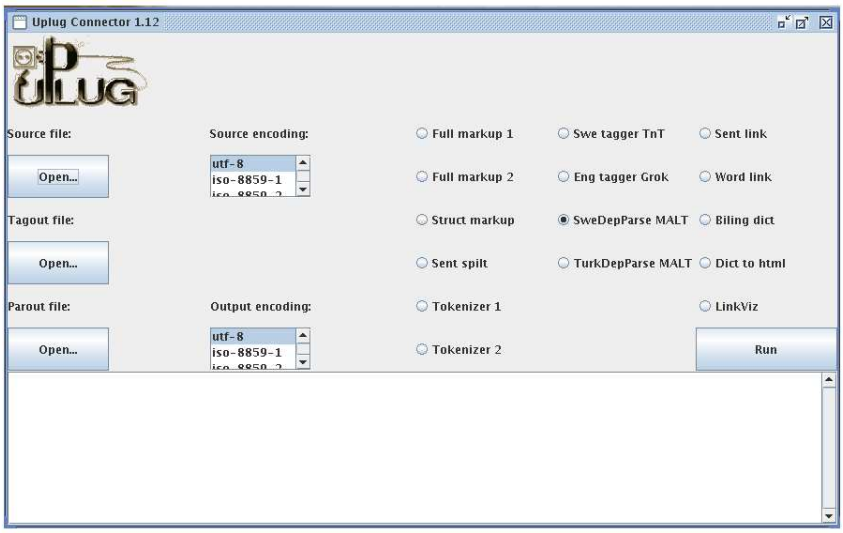


Figure 2: Uplug Connector

# References


[1] Torsten Brandts. Tnt - a statistical part-of-speech tagger. Technical report, Saarland University, 2000.

[2] J. Eryiğit G., Nivre and K. Oflazer. The incremental use of morphological information and lexicalization in data-driven dependency parsing. In *Proceedings of the 21st International Conference on the Computer Processing of Oriental Languages*, Sentosa, Singapore, December 2006.

[3] Wolfgang Lezius. Tigersearch - ein suchwerkzeug für baumbanken. In Stephan Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*, 2002.

[4] Beata Megyesi. *Data-Driven Syntactic Analysis - Methods and Applications for Swedish*. PhD thesis, Kungliga Tekniska Högskolan, 2002.

[5] Beata B. Megyesi and Bengt Dahlqvist. The swedish-turkish parallel corpus and tools for its creation. In *Proceedings of the 16th Nordic Conference of Computational Linguistics, Nodalida*, pages 136–143, Tartu, Estonia, 2007.

[6] Joakim Nivre and Johan Hall. Maltparser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories*, pages 137–148, Barcelona, December 2005.

[7] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225, 2006.

[8] Kemal Oflazer. Two-level description of turkish morphology. In *Literary and Linguistic Computing*, volume 9(2). 1994.

[9] Jörg Tiedemann. *Recycling Translations - Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing*. PhD thesis, Department of Linguistics and Philology, Uppsala University, Uppsala, 2003.

[10] Deniz Yüret and Ferhan Türe. Learning morphological disambiguation rules for turkish. In *Proceedings of HLT NAACL'06*, pages 328–334, New York, 2006.

# A The Swedish annotation script (sweparse.sh)

```
#!/bin/bash

##############################################################################################
### This script takes an input file in Uplug XML format, and performs                      ###
### part-of-speech tagging and syntactic annotation, using existing tools                  ###
###                                                                                        ###
### The script requires five arguments:                                                    ###
###                                                                                        ###
###    1) an input file in the Uplug XML format                                            ###
###                                                                                        ###
###    2) the name of the tagged output file                                               ###
###                                                                                        ###
###    3) the name of the parsed output file                                               ###
###                                                                                        ###
###    4) the encoding for the input file                                                  ###
###                                                                                        ###
###    5) the encoding for the output file                                                 ###
###                                                                                        ###
### Three output files are produced:                                                       ###
###                                                                                        ###
###    1) output file with part-of-speech tags                                             ###
###       The name of this file is specified by the user as the second argument            ###
###       when running the script                                                          ###
###                                                                                        ###
###    2) output file with part-of-speech tags and dependency relations, XCES format       ###
###       The name of this file is specified by the user as the third argument             ###
###       when running the script                                                          ###
###                                                                                        ###
###    3) output file with part-of-speech tags and dependency relations, Tiger XML         ###
###       The name of this file will be the same as for the XCES format version,           ###
###       except for the ending, that will be "_tiger"                                     ###
###                                                                                        ###
### usage:                                                                                 ###
### sh sweparse.sh  inputfile.xml  taggedoutputfile.xml  parsedoutputfile.xml              ###
###                 inputencoding  outputencoding                                          ###
###                                                                                        ###
##############################################################################################

### Read the arguments
INPUT=$1
TAGGEDOUTPUT=$2
PARSEDOUTPUT=$3
INENCODING=$4
OUTENCODING=$5
TIGERPARSEDOUTPUT=$3"_tiger"

### Set the directory where output files will be stored.
### This is assumed to be the fourth line of the file Uplugconn.cfg.
```

```
### Uplugconn.cfg should be located in the same directory where you
### run this script.
WORKINGDIR=‘head -n4 Uplugconn.cfg | tail -n1 | sed ’s/\/$//’ | sed ’s/\n//’‘

### Set the directory where the TnT-tagger is located.
### This is assumed to be the last line of the file Uplugconn.cfg.
### Uplugconn.cfg should be located in the same directory where you
### run this script.
TNTDIR=‘tail -n1 Uplugconn.cfg | sed ’s/\n//’‘

### Recode to latin1
echo ’Recoding input file to iso-8859-1...’
LANG=sv_SE recode $INENCODING..l1 < $INPUT > $INPUT.l1

### Convert the file to an appropriate format for the TnT-tagger
echo ’Converting to TnT format...’
cat $INPUT.l1 | perl -e ’while(<>){$_=˜s/\<[ˆ\>]+\>//g;print $_;}’ > $INPUT.tok

### Run the TnT tagger
echo ’Running TnT...’
$TNTDIR/tnt $WORKINGDIR/helasucx-tnt $INPUT.tok > $INPUT.tagged

### Insert the resulting tags in the original Uplug XML file
echo ’Inserting the tags in the Uplug XML format...’
LANG=sv_SE perl $WORKINGDIR/putTagsInXml.pl $INPUT.tagged $INPUT $OUTENCODING >
$INPUT.suctagged

### Recode to output encoding
echo ’Recoding tagged file to output encoding...’
LANG=sv_SE recode $INENCODING..$OUTENCODING < $INPUT.suctagged > $TAGGEDOUTPUT

### Convert to Malt-TAB format
echo ’Converting to Malt-TAB format...’
perl $WORKINGDIR/convert_tagformat.pl $INPUT.tagged > $INPUT.malttab

### Convert from SUC tags to Granska tags
echo ’Converting from SUC tags to Granska tags...’
$WORKINGDIR/remap < $INPUT.malttab > $INPUT.granska

### Fix tags that the remap-program couldn’t handle
perl $WORKINGDIR/postconvert_tagformat.pl $INPUT.granska > $INPUT.postconvert

### Run the parser with Tiger XML as output format
echo ’Parsing Tiger XML...’
$WORKINGDIR/maltparser -f $WORKINGDIR/swe/option_tiger.dat -i $INPUT.postconvert
-o $INPUT.tigerconll

### Recode from latin1 to utf-8
echo ’Recoding from latin1 to utf-8...’
LANG=sv_SE recode l1..utf-8 < $INPUT.postconvert > $INPUT.granska
```

```
#### Run the parser with Malt-TAB as output format
echo 'Parsing Malt-TAB...'
$WORKINGDIR/maltparser -f $WORKINGDIR/swe/option_tab.dat -i $INPUT.granska
-o $INPUT.conll

### Insert the resulting dependency relations in the tagged Uplug XML file
echo 'Inserting the dependency relations in the Uplug XML format...'
LANG=sv_SE perl $WORKINGDIR/putDepsInXml.pl $INPUT.conll $INPUT.suctagged $OUTENCODING
> $INPUT.parsed

### Recode to output encoding
echo 'Recoding parsed XCES file to output encoding...'
LANG=sv_SE recode utf-8..$OUTENCODING < $INPUT.parsed > $PARSEDOUTPUT

echo 'Recoding parsed Tiger XML file to output encoding...'
LANG=sv_SE recode utf-8..$OUTENCODING < $INPUT.tigerconll > $TIGERPARSEDOUTPUT

### Remove temporary files
echo 'Removing temporary files...'
rm -f $INPUT.l1
rm -f $INPUT.tok
rm -f $INPUT.tagged
rm -f $INPUT.suctagged
rm -f $INPUT.malttab
rm -f $INPUT.granska
rm -f $INPUT.postconvert
rm -f $INPUT.conll
rm -f $INPUT.tigerconll
rm -f $INPUT.parsed

echo 'Output files are in: '
echo $TAGGEDOUTPUT
echo $PARSEDOUTPUT
echo $TIGERPARSEDOUTPUT
```

# B The Turkish annotation script (turparse.sh)

```
#!/bin/bash

#############################################################################################
### This script takes an input file in Uplug XML format, and performs                     ###
### part-of-speech tagging and syntactic annotation, using existing tools                 ###
###                                                                                       ###
### The script requires five arguments:                                                   ###
###                                                                                       ###
###    1) an input file in the Uplug XML format                                           ###
###                                                                                       ###
###    2) the name of the tagged output file                                              ###
###                                                                                       ###
###    3) the name of the parsed output file                                              ###
###                                                                                       ###
###    4) the encoding for the input file                                                 ###
###                                                                                       ###
###    5) the encoding for the output file                                                ###
###                                                                                       ###
### Three output files are produced:                                                      ###
###                                                                                       ###
###    1) output file with part-of-speech tags                                            ###
###       The name of this file is specified by the user as the second argument           ###
###       when running the script                                                         ###
###                                                                                       ###
###    2) output file with part-of-speech tags and dependency relations, XCES format      ###
###       The name of this file is specified by the user as the third argument            ###
###       when running the script                                                         ###
###                                                                                       ###
###    3) output file with part-of-speech tags and dependency relations, Tiger XML        ###
###       The name of this file will be the same as for the XCES format version,          ###
###       except for the ending, that will be "_tiger"                                    ###
###                                                                                       ###
### usage:                                                                                ###
### sh turparse.sh  inputfile.xml  taggedoutputfile.xml  parsedoutputfile.xml             ###
###                 inputencoding  outputencoding                                         ###
###                                                                                       ###
#############################################################################################

### Read the arguments
INPUT=$1
TAGGEDOUTPUT=$2
PARSEDOUTPUT=$3
INENCODING=$4
OUTENCODING=$5
TIGERPARSEDOUTPUT=$3"_tiger"

### Set the directory where output files will be stored.
### This is assumed to be the fifth line of the file Uplugconn.cfg.
```

```
### Uplugconn.cfg should be located in the same directory where you
### run this script.
WORKINGDIR=‘head -n5 Uplugconn.cfg | tail -n1 | sed ’s/\/$//’ | sed ’s/\n//’‘

### Recode to latin5
echo ’Recoding input file to iso-8859-9...’
LANG=tr_TR recode $INENCODING..l5 < $INPUT > $INPUT.l5

### Convert the file to the appropriate format for the script
### prepareconllfromsentence.pl:
###    - Remove XML-tags
###    - Remove empty lines
###    - Separate sentences with *******
echo ’Converting to appropriate format...’
LANG=tr_TR perl $WORKINGDIR/reformat.pl < $INPUT.l5 > $INPUT.reformat

### Perform morphological analysis and disambiguation.
### Output in a format appropriate for the MALT parser.
echo ’Morphological analysis and disambiguation...’
LANG=tr_TR perl $WORKINGDIR/prepareconllfromsentence.pl $INPUT.reformat $WORKINGDIR

### Insert the resulting tags in the original Uplug XML file
echo ’ ’
echo ’Inserting the tags in the UPLUG XML format...’
LANG=tr_TR perl $WORKINGDIR/putTagsInXml.pl sample-output.txt $INPUT $OUTENCODING >
$INPUT.tagged

### Recode to output encoding
echo ’Recoding tagged file to output encoding...’
LANG=tr_TR recode $INENCODING..$OUTENCODING < $INPUT.tagged > $TAGGEDOUTPUT

### Recode from latin5 to utf-8 before running the parser
echo ’Recoding from latin5 to UTF-8...’
LANG=tr_TR recode l5..utf-8 < $INPUT.reformat.postagged > $INPUT.postagged

### Make sure that the format of the file i OK, before running the parser
echo ’Checking the format...’
cat $INPUT.postagged | LANG=tr_TR egrep -v ’DERIV’ | LANG=tr_TR perl -e
’$count=0;while(<>){$_=~s/A3sg\tA3sg\tPnon\|Nom\t/Noun\tNoun\tA3sg\|Pnon\|Nom\t/;
$count++;$_=~s/^(\s*)\d+(\s+)/$1$count$2/;print $_;}’ > temp

#### Run the parser with Malt-TAB as output format
echo ’Parsing Malt-TAB...’
LANG=tr_TR $WORKINGDIR/maltparser -f $WORKINGDIR/tur/option_tab.dat
-i $INPUT.postagged -o $INPUT.conll.norecode

### Run the parser with Tiger XML as output format
echo ’Parsing Tiger XML...’
LANG=tr_TR $WORKINGDIR/maltparser -f $WORKINGDIR/tur/option_tiger.dat -i temp
-o $INPUT.tigerconll.norecode
```

```
### Recode to output encoding
echo 'Recoding parsed XCES file to output encoding...'
LANG=tr_TR recode utf-8..$OUTENCODING < $INPUT.conll.norecode > $INPUT.conll

### Make sure that the morphological information is preserved in the Tiger XML file
echo 'Converting the postags into full postags...'
LANG=tr_TR cut -f5,6 $INPUT.conll.norecode | LANG=tr_TR egrep -v '^\s*$' > morphtags.txt
LANG=tr_TR perl $WORKINGDIR/expand_postags.pl morphtags.txt $INPUT.tigerconll.norecode
> tmp

### Recode to output encoding
echo 'Recoding parsed Tiger XML file to output encoding...'
LANG=tr_TR recode utf-8..$OUTENCODING < tmp > $TIGERPARSEDOUTPUT

### Insert the resulting dependency relations in the tagged Uplug XML file
echo 'Inserting the dependency relations in the UPLUG XML format...'
LANG=tr_TR perl $WORKINGDIR/putDepsInXml.pl $INPUT.conll $TAGGEDOUTPUT $OUTENCODING >
$PARSEDOUTPUT

### Remove temporary files
echo 'Removing temporary files...'
rm -f $INPUT.l5
rm -f $INPUT.reformat
rm -f $INPUT.tagged
rm -f $INPUT.reformat.postagged
rm -f $INPUT.postagged
rm -f $INPUT.conll.norecode
rm -f $INPUT.tigerconll.norecode
rm -f $INPUT.conll
rm -f $INPUT.parsed
rm -f output.txt
rm -f sample-output.txt
rm -f temp.dat
rm -f temp.txt
rm -f tmp
rm -f temp
rm -f notf.dat
rm -f morphtags.txt

echo 'Output files are in: '
echo $TAGGEDOUTPUT
echo $PARSEDOUTPUT
echo $TIGERPARSEDOUTPUT
```