M A T E M A T I C K O - F Y Z I K Á L N Í F A K U L T A

P R A H A

DEPFIX MANUAL

RUDOLF ROSA

ÚFAL Technical Report
TR-2014-55

ISSN 1214-5521

U N I V E R S I T A S C A R O L I N A P R A G E N S I S

Copies of ÚFAL Technical Reports can be ordered from:

Institute of Formal and Applied Linguistics (ÚFAL MFF UK)
Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, CZ-11800 Prague 1
Czech Republic

or can be obtained via the Web: http://ufal.mff.cuni.cz/techrep

Depfix Manual

Rudolf Rosa,
rosa@ufal.mff.cuni.cz
Charles University in Prague,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, Prague, CZ-11800, Czechia

September 8, 2014

Abstract

The manual gives instructions on installing and running Depfix, our open-source tool for automatic post-editing of machine translation. We cover the steps required to use Depfix to process your own data. The manual also contains a description of the Depfix pipeline, including instructions that will enable you to modify the operation of Depfix.

Introduction

Depfix is a tool for automatic post-editing of outputs of English-to-Czech machine translation (MT), especially phrase-based statistical machine translation (PB-SMT). Depfix uses a pipeline of natural language processing (NLP) tools to analyze the input sentences, such as taggers and parsers. The corrections Depfix performs are mainly rule-based, and rely heavily on the analyses provided by the tools.

Depfix was introduced in [1], and subsequent improvements were described especially in [8] and [9]. In [5], Depfix has been released as an open source tool, under the GNU GPL v2 licence. For a comprehensive description of the whole Depfix system, please refer to [4].

In this manual, we give detailed instructions on installing and running Depfix, including a documentation of all commands and settings. We also give a general overview of the structure of the source codes, which may serve as a starting point for modifying them, or for using them for inspiration in your own post-editing system. Brief instructions are also bundled directly with Depfix in a `README` file.

Contents

1 System requirements

You need to have at least 3.5 GB of RAM to run the default version of Depfix. So far, Depfix has only been tested on Linux. In principle, it should be possible to run Depfix on other operating systems as well, such as Microsoft Windows with Cygwin[1] – however, this has not been tested, and we can provide little or no support in that area.

2 Installing Depfix

2.1 TL;DR

```
wget -O install.tgz http://goo.gl/FpIOuw
tar -zxvf install.tgz
cd install
bash all.sh
```

Please note that running the installation script will take tens of minuted or even hours, and some of the steps may occasionally ask you some questions – so you should leave the script running but check it time to time.

If this works for you: congratulations, you can now skip to a test run of Depfix – see Section 2.9 (you may try to do the test run even if the installation prints out error messages, as these may be non-fatal). Otherwise, you may want to inspect the error messages you get and try to find a solution yourself, or you may refer to the following sections.

2.2 Introduction

Depfix is implemented in Treex framework and is contained in the Treex Subversion repository,[2] in the `trunk/treex/devel/depfix` directory. Thus, to install Depfix, please refer to the Treex installation manual at `http://ufal.mff.cuni.cz/treex/install.html` and follow the steps 1 (Perl), 2 (Treex::Core), 3 (Treex::EN), and 5 (SVN). Step 4 (TrEd) is optional; installing TrEd will allow you to inspect intermediate Depfix files that contain linguistic analyses of the sentences, but is not required if you are happy with only using the input and output, which is plaintext. Step 6 (Featurama) is not needed for Depfix.

There are several additional steps that are typically required when installing Treex. For ease of use, we list here the whole procedure that currently seems to work for Ubuntu 14.04. Still, please note that installing Treex is a complex task, and you may need to perform the steps differently on your machine. Also, some steps may not be necessary in your situation, such as installing packages that you have already installed.

All of the codes listed in the following section can be downloaded as bash scripts from `http://ufallab.ms.mff.cuni.cz/~rosa/depfix/install.tar.gz`, e.g. in the following way:

```
wget -O install.tgz http://goo.gl/FpIOuw
tar -zxvf install.tgz
cd install
```

[1] `https://www.cygwin.com/`

[2] `https://public:public@svn.ms.mff.cuni.cz/projects/tectomt_devel/`

Using the downloaded scripts may be more reliable then copy-pasting the codes from the PDF file.

2.3 Installing missing packages

We recommend installing several software packages before installing Treex.

```
sudo apt-get install subversion libxml2-dev zlib1g-dev g++ \
cmake libboost-all-dev
```

2.4 Installing Treex

This is just a very short version of the full Treex installation manual. If this works for you, then fine. If it doesn't, please refer to the original manual at `http://ufal.mff.cuni.cz/treex/install.html`.

```
# Prepare Perl environment
wget -O- http://cpanmin.us | \
 perl - -l ~/perl5 App::cpanminus local::lib
eval `perl -I ~/perl5/lib/perl5 -Mlocal::lib`
echo '## Treex installation ##'                          >> ~/.bashrc
echo 'eval `perl -I ~/perl5/lib/perl5 -Mlocal::lib`'     >> ~/.bashrc
grep bashrc ~/.bash_profile || echo 'source ~/.bashrc' \
 >> ~/.bash_profile
# Install part of Treex that is on CPAN
cpanm --force Treex::Core Treex::EN
# Check out current version of full Treex from its SVN repository
# Password is "public"
SVN_TRUNK=https://svn.ms.mff.cuni.cz/svn/tectomt_devel/trunk
svn --username public co $SVN_TRUNK/treex ~/treex
svn --username public co $SVN_TRUNK/libs/other ~/treex/oldlib
echo export PATH="$HOME/treex/bin:$PATH" >> ~/.bashrc
echo export \
 PERL5LIB="$HOME/treex/lib:$HOME/treex/oldlib:$PERL5LIB" \
 >> ~/.bashrc
echo export TMT_ROOT=$HOME/.treex >> ~/.bashrc
source ~/.bashrc
```

2.5 Installing missing CPAN modules

We recommend installing several CPAN modules:

```
cpanm --force forks Lingua::Interset POE Ufal::MorphoDiTa \
 App::whichpm Class::Std Locale::Language String::Util \
 MooseX::Role::AttributeOverride PerlIO::gzip
```

The list of the modules may change as Treex is further developed, so please install any other missing modules. Whenever you get an error message such as `Can't locate Ufal/MorphoDiTa.pm in @INC` (followed by many lines of garbage you can ignore), use the `cpanm` installer in the following way to obtain it:

```
cpanm Ufal::MorphoDiTa
```

In some cases, you may have to use the `--force` switch to overcome some errors – for example, some of the packages seems to have errors in tests, which leads to errors even if the package installs successfully.

2.6 Installing Treex models

Many parts of Depfix require model files to run. There is a set of models that are required in the current version and seem not to be downloaded automatically.[3] Thus, we recommend downloading them beforehand:

```
for m in \
mst_parser/cs/pdt20_train_autTag_golden_latin2_pruned_0.02.model \
morpho_analysis/cs/b2800a.f2o \
morpho_analysis/cs/config_a.cfg \
morpho_analysis/cs/config_g.cfg \
morpho_analysis/cs/CZ100404ac.txt \
morpho_analysis/cs/CZ100404ad.in \
morpho_analysis/cs/CZ100404ad.out \
morpho_analysis/cs/CZ100404ae.cpd \
morpho_analysis/cs/CZ100404af.sgm \
morpho_analysis/cs/CZ100404ag.txt \
morpho_analysis/cs/CZ100404ak \
morpho_analysis/cs/CZ100404am.x \
morpho_analysis/cs/CZ100404at.x \
morpho_analysis/cs/CZ100404au.cpd \
morpho_analysis/cs/CZ100404aw.cpd \
morpho_analysis/cs/CZ100404ax \
morpho_analysis/cs/CZ100404ax.README \
morpho_analysis/cs/gmon.out \
morpho_analysis/cs/hgddCZ.cpd \
morpho_analysis/cs/morfo_a.log \
morpho_analysis/cs/morfo_g.log \
morpho_analysis/cs/unhandled_a.log \
morpho_analysis/cs/unhandled_g.log
do
wget -O - \
http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/$m \
| install -D /dev/stdin ~/.treex/share/data/models/$m
done
```

Whenever Treex dies because it cannot find a required model file, it will report the path to the missing file; the error message will look something like this:

```
Cannot find ~/.treex/share/data/models/some/model
```

In such case, please download the missing file from:

```
http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/some/model
```

Probably the easiest way to do that is using wget:

```
wget -O - \
http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/some/model \
| install -D /dev/stdin ~/.treex/share/data/models/some/model
```

2.7 Installing MGiza++

You may need to install the MGiza++ word-aligner if this is indicated by error messages. To do so, you should visit the webpage of MGiza++, `http://sourceforge.net/projects/mgizapp/`, and follow the instructions. Currently, the easiest way to install MGiza++ seems to be the following:

[3] Treex supports automatic download of missing files, but it may fail for various reasons.

```
# Use any directory you wish (and have write access to).
MGIZA_DIR=$HOME/mgizapp-code
svn checkout http://svn.code.sf.net/p/mgizapp/code/trunk $MGIZA_DIR
cd $MGIZA_DIR/mgizapp/
cmake .
make
make install
```

Next, you need to ensure that Treex finds your MGiza++ installation:

```
mkdir -p ~/.treex/share/installed_tools/mgizapp
ln -s $MGIZA_DIR/mgizapp/inst  \
~/.treex/share/installed_tools/mgizapp/install
```

When running Depfix, it may happen that you will get an error message that looks like this:

```
~/.treex/share/installed_tools/mgizapp/install contains binaries
compiled for 'x86_64-linux' but you are using
'x86_64-linux-thread-multi'
```

Most likely, if you edit the Treex wrapper for MGiza++ (`Treex::Block::Align::A::MGiza`) and disable the architecture check, MGiza++ will run just fine. To do so, open `~/treex/lib/Treex/Block/Align/A/AlignMGiza.pm` in a text editor (such as vim, emacs or gedit), and delete the line which looks like this (in current version it is line 34):

```
log_fatal "$mgizadir␣contains␣binaries␣compiled␣for␣'x86_64-linux...
```

2.8 Installing CzechMorpho

You may need to install CzechMorpho if this is indicated by error messages. This is a Perl module which is not present on CPAN, so you will have to run the following commands to install it:

```
svn export $SVN_TRUNK/libs/packaged/CzechMorpho/ /tmp/CzechMorpho/
cd /tmp/CzechMorpho/
perl Build.PL
./Build
./Build test
./Build install
```

2.9 Testing run of Depfix

Once you have everything installed, go to the `treex/devel/depfix` directory of the SVN checkout using the command-line:

```
cd ~/treex/devel/depfix
```

Depfix is operated using `make` targets (which are called *commands* in this document). To test if you have installed Depfix correctly and to download several required models, invoke the `test` command:

```
make test
```

This will try to run Depfix on 1 testing sentence and will take about 5 minutes to run, plus time to download required files (several hundreds of MB).

When it finishes, the end of the standard output should look approximately like that (the lines have been shortened here to fit on page):

```
DEPFIX [2014-08-27_20-44-21_2952406743_small_test]: eval_show ...
chgd    NIST_0  NIST_1  diff    BLEU_0  BLEU_1  diff ...
1       2.5483  3.2433  69.49   0.3349  0.4625  12.76 ...
make[1]: Leaving directory '/home/rur/treex/devel/depfix'
```

If the test ends successfully, printing out BLEU and other scores, you have Depfix installed and running. If not, please refer to the preceding steps, the Treex installation manual, your common sense, the Internet, and, if nothing helps, feel free to contact us (see Section 5).

3 Running Depfix

3.1 The basics

You can run the whole Depfix pipeline by invoking the default target:

```
make
```

This is equivalent to invoking all the commands corresponding to main Depfix processing steps:

```
make init totreex tag run_giza ner_en parse \
fix deepfix write_sentences eval
```

Without any settings, this is very similar to running `make test`, as the default input to Depfix is the 1 testing sentence.

The `init` command creates a new experiment directory and copies the input files into it. It is therefore wise to specify several settings when invoking it (`Makefile` variables are used for this):

DIRLABEL the label to use for the new experiment directory

DATA_EN the source English data, i.e. the *input* of the SMT system

DATA_CS the Czech data to be processed, i.e. the *output* of the SMT system

REFERENCE_CS the reference Czech translation to be used for evaluation[4]

The format of the input data is plaintext in UTF-8 encoding, one sentence per line. Depfix expects the sentences to be in "human" format (e.g. not lowercased or tokenized).

Depfix comes with a sample WMT10 dataset, so you can run the following command to initialize a new experiment directory with that dataset:

```
make init DIRLABEL=wmt10_test DATA_CS=data/wmt10_bojar-2012.cs \
DATA_EN=data/wmt10_src.en REFERENCE_CS=data/wmt10_ref.cs
```

The output of the `init` command will look something like this:

[4]Currently, Depfix requires that `REFERENCE_CS` is provided, so if you do not have the reference translation, fill in the same value as for `DATA_CS`.

```
DEPFIX [2014-09-08_11-38-38_1255015118_wmt10_test]: init
  REFERENCE_CS=data/wmt10_ref.cs DATA_EN=data/wmt10_src.en
  DATA_CS=data/wmt10_bojar-2012.cs
Working directory:

2014-09-08_11-38-38_1255015118_wmt10_test
```

A new working directory is generated for the experiment (and the input files are copied into it). The name of the directory is printed to the standard output, as shown in the example above.[5]

The next step is running Depfix on this directory. As you have already invoked the `init` target, you now need to run all but the first step. It is of course possible to list the steps manually, but an easier way is to use the `default_ni` target, which is identical to the default target except for not containing the `init` step (its name stands for "default, no init").

Whenever you are invoking a target which does not create a new experiment directory, but uses an already existing experiment directory, you must always specify the experiment directory using the `DIRNAME` setting.

Thus, to run Depfix on the experiment directory created in the previous step, it suffices to use:

```
make default_ni DIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

The sample WMT10 dataset has 2489 sentences; expect Depfix to run for about 2 hours on it. Once it is finished, it will print our automatic evaluation results and exit.

Depfix output can be found in the `output.txt` file in the experiment directory, or displayed using the `view` target:

```
make view DIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

To see what has been changed by Depfix, use the `compare` target (or `compare_log` to also see details about the fixes applied):

```
make compare DIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

There are many other commands available, although the already mentioned ones are sufficient for basic operation of Depfix. For a full documentation, see Section 3.3.

There are also many settings available, most of them specifying which Treex scenario should be used for each processing step. Two more are detailed in the next section; the other can usually be left at their default values. For a full documentation, see Section 3.4.

3.2 Running the full version of Depfix

By default, only a basic version of Depfix is run. The full state-of-the-art version differs from the basic version in several aspects:

- it uses a parser adapted to parsing SMT outputs
- it uses a statistical fixing component

[5] The experiment directory name format is `y-m-d_h-m-s_random_DIRLABEL`.

- it requires about 20 GB of RAM to run
- it runs about twice as long as the basic version
- it achieves slightly better results – e.g. on the sample WMT10 data, applying the full version leads to an improvement of 0.45 BLEU point, while the basic version achieves a 0.37 BLEU point improvement

The parser used by default for Czech is a version of the Maximum Spanning Tree (MST) parser [2], adapted for standard Czech [3], but not for SMT outputs. If you have more than 20 GB of RAM, you should use the MSTperl parser adapted for SMT outputs [7, 6] instead, which is more robust to various grammatical errors commonly occurring in SMT outputs, and uses additional features that have shown to be useful in that setting (such as features based on the source English sentence parse tree). To switch to this parser, use the following additional setting when running Depfix (more specifically, when invoking the `parse` target):

```
CS_ANALYSIS_2_SCEN=cs_analysis_2_boost_model_025.scen
```

You can also enable the statistical fixing component [9], which targets errors in valency of verbs and nouns. It requires at least 15 GB RAM to run. We do use it in our setups, but please note that evaluation of the effect of using the component has brought inconclusive results – it usually improves the results, but only very slightly. To enable the component, use the following setting (relevant for the `deepfix` target):

```
TFIX_SCEN=tfix_cut_ChgCase2.scen
```

If you want to use full Depfix by default, you can make these (as well as any other) settings default by using the `settings.mak` file; instructions are included in the file itself (edit the file using a plaintext editor).

3.3 Depfix commands

Depfix commands are Makefile targets – you may inspect them by viewing the source code of `depfix/Makefile`. This section documents the current set of available commands. Please note that Depfix is an experimental software in active development, and commands may be added, removed or modified; also, some commands may become temporarily unusable.

Most of the commands are parametrized with variables, listed in the following section. In this section, we give the names of the variables with their default values wherever applicable – e.g. `OUTPUT_TXT=output.txt` refers to a variable called `OUTPUT_TXT` whose default value is `output.txt`.

As already mentioned, the most important variable is `DIRNAME`, which specifies the working directory to be used, and must be set for any command that does not create a new directory. **Do not set** `DIRNAME` if you invoke the `default` command, the `init` command, or the `clone` command. For any other target, you **must set** `DIRNAME` – e.g.:

```
make view DIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

Most of other variables specify scenarios to use for the commands – their value is always a name of a file in the `depfix/scenarios` subdirectory – or filenames

to use. You do not have to set these variables unless you want to modify the way Depfix behaves, their default values are just fine.

While running, Depfix prints brief information to the standard output, listing the command currently being run (including subcommands), the experiment directory, and some of the settings. More detailed information is printed to the standard error output.

3.3.1 Main Depfix pipeline

The main Depfix pipeline is invoked by the `default` command, which invokes a set of subcommands that correspond to individual steps of the processing pipeline. Invoking those commands explicitly is useful if you need to run only a part of the pipeline – for example, if you want to play with the fixing rules, you may run the analysis up to the `parse` step, make several copies of the experiment directory, and then run the rest of the pipeline on each of the directories separately with different fixing scenarios.

At end of the `parse` step, the intermediate files are backed-up in a `parsed` subdirectory of the working directory, and they are restored at beginning of the `fix` step. Thus, if you run the `fix` step on an already fixed directory, the intermediate files will first be restored to the state in which they were after the `parse` step. This is generally not true for the other steps, so running them more than once on one directory may lead to errors. Still, if a step dies, it is *often* safe to try to run it again – e.g. if the pipeline dies while performing the `run_giza` step, it is worth trying to rerun the pipeline again by explicitly listing all the steps starting at `run_giza`.

default Run the whole Depfix pipeline on the input data (see `init` for important variables to set). Typing `make` without specifying any command will also invoke the `default` command. `default` invokes the following commands: `init totreex tag run_giza ner_en parse fix deepfix write_sentences eval`

default_ni Identical to `default` except for not invoking the `init` command. The `DIRNAME` variable must be specified. To be used if you prefer to init your working directory in a separate step, and then run Depfix on it later.[6]

init Create a new experiment directory and copy input data into it. Set a name for the new directory using `DIRLABEL`.[7] Specify the input data using `DATA_CS`, `DATA_EN` and `REFERENCE_CS`; the value of each of these variables must be a path to a text file (UTF-8, one sentence per line). The default values correspond to the sample 1-sentence input used in `test` (`DATA_CS=data/csw.txt DATA_EN=data/en.txt REFERENCE_CS=data/cs.txt`); to use a larger sample input, you may use the following setting:

```
DATA_CS=data/wmt10_bojar-2012.cs
DATA_EN=data/wmt10_src.en
REFERENCE_CS=data/wmt10_ref.cs
```

[6] This is quite a common practice – you may want to prepare your experiment directories by running `init` fully manually, as this involves setting a lot of variables and takes very little time to run, and then let run `default_ni` without paying attention to it in parallel for several experiments, as this can be fully automated and usually takes hours.

[7] The experiment directory name format is `y-m-d_h-m-s_rand_DIRLABEL` and the name of the newly created directory, together with other information, is printed to standard output.

The reference Czech translation is only used in the last step – it is not analyzed by NLP tools and is not used in the fixing.

totreex Read the input text data into Treex files and perform tokenization projection. The resulting files are called `translation001.treex.gz`, `translation002.treex.gz`, etc., and each contains 100 sentences (or, more precisely, 100 source English sentences + 100 Czech machine translation sentences + 100 Czech reference translation sentences). The following steps operate on these files – each step reads them from disk when it starts and saves them modified back to disk when it ends. The files are in PML, which is an XML format, and can be viewed by the TrEd editor. The tokenization projection is performed by the scenario specified in `PROJECT_TOKENIZATION_SCEN=project_tokenization.scen`.

tag Tokenize the sentences and run morphological analyses on them. (`EN_ANALYSIS_1_SCEN=en_analysis_1.scen`, `CS_ANALYSIS_1_SCEN=cs_analysis_1_morphodita.scen`). After that step, each sentence is represented by a set of nodes, corresponding to the tokens, and each of the nodes has the following three attributes filled: `form`, `lemma`, and `tag`. The tokens are contained in an a-tree (although the tree is flat at this moment – all nodes are children of the technical root node).

run_giza Word-align the sentences (`RUN_GIZA_SCEN=run_mgiza.scen`) and run heuristics to add missing alignment links (`ADD_MISSING_LINKS_SCEN=add_missing_links.scen`). The word-alignment used is of the intersection type, i.e. for each token there is at most one aligned token. The alignment links are stored in attributes of the English nodes.[8]

ner_en Run named entity recognizer for English (`NER_EN_SCEN=ner_en.scen`). This creates an n-tree which contains named entities found in the sentence.

parse Run a dependency parser on the sentences (`EN_ANALYSIS_2_SCEN=en_analysis_2.scen`, `CS_ANALYSIS_2_SCEN=cs_analysis_2_msta.scen`). After this step, the a-trees created in the `tag` step are no more flat but instead represent the dependency structure of the sentence. This step consists of two substeps: `parse_only`, which does the parsing, and `parse_backup`, which copies the treex files into the `parsed` subdirectory.

fix Run the Depfix fix rules. This step consists of four substeps. First, `restore_parsed` replaces the current treex files by the files from the `parsed` subdirectory. Next, `fix_prepare` prepares the files for Depfix by copying the a-trees into a new `T` zone, so that the files will contain both the original and the fixed sentences for easy observation of the fixes performed (`FIX_PREPARE_SCEN=fix_prepare.scen`). After that, `ner_cs` runs a named entity recognizer for Czech (`NER_CS_SCEN=ner_cs.scen`). And finally, `fix_run` runs the actual fix rules (`FIX_SCEN=fix.scen`).

deepfix Run the fixes that operate on t-layer. This command consists of four subcommands. `a2t` creates a t-layer analysis (`A2T_CS_SCEN=a2t_cs.scen`,

[8] However, this is not very practical for operation of Depfix, which operates on the Czech nodes. Therefore, the alignment will be stored in attributes of the Czech nodes in near future.

A2T_EN_SCEN=a2t_en.scen). tfix_prepare adds word alignment links from a-layer (TFIX_PREPARE_SCEN=tfix_prepare.scen). tfix performs the t-layer fixes (TFIX_SCEN=tfix_rules.scen); the fixes are immediately projected to a-layer. refix_after_tfix runs the a-layer fixes again especially in order to fix agreement that may have been violated while performing t-fixes (REFIX_SCEN, by default identical to FIX_SCEN).

write_sentences Write the fixed sentences into OUTPUT_TXT=output.txt (WRITE_SENTENCES_SCEN=write_sentences.scen). Also calls a write_fixlog substep that writes a log containing the fixes performed into fixlog.txt.

eval Perform automatic evaluation, storing the result into AUTOEVAL_OUT=autoeval.out. This step consists of several substeps, each of which computes a different characteristic and stores it. You may specify a different reference translation file to be used (REF_CS_TXT=ref_cs.txt). eval_dirname stores the name of the working directory. eval_lines computes the number of sentences modified by Depfix. eval_bleu computes BLEU and NIST scores. eval_ter computes PER and TER scores. eval_show processes the contents of the AUTOEVAL_OUT file and prints out the results formatted as tab separated values.

3.3.2 Viewing the results of Depfix run

Most of the commands in this section can be parametrized using the following variables, specifying the names of files in the experiment directory to use (but typically the default values, given in brackets, are what you want, so you usually do not need to set these):

ORI Original Czech translation (data_cs.txt)

NEW Depfix output (OUTPUT_TXT)

REF Reference Czech translation (REF_CS_TXT)

EN Source English sentences (data_en.txt)

The names of these variables are also used as labels for the sentences in the compare* commands.

The following commands are available for displaying the results of Depfix in various ways:

view Show the output of Depfix, stored in file specified by OUTPUT_TXT. Only the resulting sentences are show, one sentence per line.

compare Show the output of Depfix together with the inputs. For each sentence, the original Czech sentence (ORI) and the fixed sentence (NEW) are compared, the differences are highlighted, and the sentences are printed together with the source English sentence (EN) and the reference Czech translation (REF). If there is no difference between the original and the fixed sentence, the sentences are not printed.

compare_log Similar to compare, but also showing a fix log, which contains a list of fixes that were applied on the sentence.

compare_ci Similar to `compare`, but the search for differences is done case-insensitively – differences that are only in casing are not highlighted, and sentences in which Depfix changed only casing are not considered as changed (and therefore not printed).

compare_log_ci Similar to `compare_log`, but the search for differences is done case-insensitively as in `compare_ci`.

comparehtml Similar to `compare`, but the output is in HTML format (differences are highlighted by bold font). The resulting HTML file is printed to standard output – use stdout redirection to save it into an HTML file.

compare2 Compare results of two experiments run on the same input data – differences between the two outputs of Depfix are highlighted, instead of differences between the input and output. The two experiment directories are to be specified by `DIRNAME` and `DIRNAME2`:

```
make compare2 \
DIRNAME=2014-07-15_17-37-21_4908631573_wmt10_test \
DIRNAME2=2014-09-08_11-38-38_1255015118_wmt10_test
```

compare2html Similar to `compare2`, but the output is in HTML format as in `comparehtml`.

3.3.3 Manual evaluation

There is a set of commands for carrying out manual evaluation of Depfix processing (`maneval_prepare`, `maneval_eval`, `cross_annot_agree`, `cross_annot_agree_matrix`, and other). As the needs for manual evaluation are typically different every time the evaluation is performed, we suggest you use these commands as inspiration when performing manual evaluation – please inspect the source code of the `Makefile` for the code used.

The commands make use of a simple manual evaluation tool by Ondřej Bojar, called QuickJudge.[9] The tool is bundled with Depfix, as it consists of one Perl script, and resides in `depfix/scripts/quickjudge.pl`. It generates text files with randomized order of outputs of various MT systems or setups, which makes blind evaluation possible. When annotators have marked in the text files which translations are better, the tool then processes the annotations, which makes it possible to compute evaluation results.

Still, if you need to perform a manual evaluation of Depfix and do not want to bother with tweaking the commands, the basic procedure that works out-of-the-box is the following:

1. Run the `maneval_prepare` command. You must set `DIRNAME`, and you may set other variables: the number of lines to be annotated (`ANNOT_LINES=3003`), the number of chunks to split these lines into (`ANNOT_NUM=20`), and the prefix to use for the newly created manual evaluation directory (`MAN_PREFIX=depfix_maneval`). So, the command you run may look like that if you want to evaluate 500 sentences by 5 annotators:

[9] `http://ufal.mff.cuni.cz/euromatrix/quickjudge/`

```
make maneval_prepare ANNOT_LINES=500 ANNOT_NUM=5 \
DIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

The manual evaluation directory will have a name following the pattern `MAN_PREFIX_DIRNAME` and will contain a number of generated files. Please note that the `maneval_prepare` command splits the files before looking at the changes, so if you e.g. want to evaluate the first 500 sentences (`ANNOT_LINES=500`), you may expect to obtain about 300 sentences for annotation, as many sentences are not changed by Depfix and there is nothing to annotate on them. For the same reason, if you split these into 5 chunks, (`ANNOT_NUM=5`), each chunk will contain a different number of sentences (e.g. the third chunk will contain sentences from 201st to 300th that were changed by Depfix).

2. Take the `*.anot` files from the manual evaluation directory and give them to your annotators with appropriate instructions. Files are to be annotated by adding any one character (e.g. 1, *, y...) to the beginning of the line which is BETTER than the other line, e.g.:

```
EN_ORIG A big role in the film is played by Matt Damon.
CS_REF  Velkou roli roli hraje herec Matt Damon.
*       Velkou roli ve filmu hraje Matt Damon.
        Velkou roli ve filmu hrajou Matt Damon.
```

Do not do anything with the line which is not the good one. If quality of the lines is equal, do not add anything anywhere, just leave it as it is. The lines start with a `TAB` character, which must be kept. The lines marked `EN_ORIG` and `CS_REF` are just for information. Do not add any newlines!

3. Put the annotated files back into the manual evaluation directory (overwrite their unannotated versions, these are not needed any more), and run `maneval_eval` (you need to specify `DIRNAME`; if you used a custom `MAN_PREFIX`, you need to specify that as well).

3.3.4 Other commands

test Run the whole Depfix pipeline on a sample input, consisting of only one sentence. You do not need to set any variables for this command, everything is already preset.

help Display the `README` file, which contains brief instructions on running Depfix.

clone Copy an experiment directory (`OLDDIRNAME`) to a new one (created according to `DIRLABEL`). Similar to performing `cp -r`, but including the generation of a new `DIRNAME` as in `init`, i.e. containing the date and time. As the `clone` command fills the name of the generated directory into the `DIRNAME` variable, it can be followed by other Depfix commands.

```
make clone DIRLABEL=experiment_2 \
OLDDIRNAME=2014-09-08_11-38-38_1255015118_wmt10_test
```

bootstrap_eval Run bootstrap resampling significance test to estimate whether the difference in BLEU score achieved by Depfix post-editing (as computed by `eval`) is statistically significant. The number of samples is set by `SAMPLES=1000`, the significance level by `ALPHA=0.05`. Use with caution, as some authors do not regard the results of bootstrap resampling as credible.

Several technical or experimental commands are not described in this manual.

3.4 Depfix variables

Depfix commands can be parametrized by using variables (these are standard Makefile variables). All of the variables are set to default values, so in most cases you will not need to set them explicitly. However, there are several variables that must be always specified:

- the variables for the `init` target, which specify the input files and the label for the new experiment directory (`DATA_CS`, `DATA_EN`, `REFERENCE_CS`, `DIRLABEL`),
- the `OLDDIRNAME` variable for the `clone` target,
- and, most importantly, the `DIRNAME` variable, which specifies the experiment directory to be used, and must be specified for any set of commands that does not start with one of the following commands: `init`, `default`, `clone`.

Several technical or experimental variables are not described in this manual – we focus on variables that may be worth changing in some situations.

3.4.1 Setting the variable values

Depfix variables may be set on the command line when invoking a Depfix command:

```
make init DIRLABEL=wmt14_bojar
```

Moreover, the default settings of the variables can be overridden by editing the `settings.mak` file in a plaintext editor. The file is loaded after the `Makefile`, so settings in `settings.mak` have priority over the defaults specified in the `Makefile`; however, you can still override these by setting the variables on the command line, which has always the highest priority.

The `settings.mak` file contains information on how to use it; generally, you set a new default value for a variable in the following way – e.g. to use a different scenario for writing out the sentences without performing detokenization:

```
WRITE_SENTENCES_SCEN=write_sentences_no_detok.scen
```

If a setting appears multiple times in the file, the last occurrence is the one that is valid.

The file also contains prepared settings for running the full version of Depfix (see Section 3.2), and for running Treex on an SGE cluster (these settings are commented out by default).

3.4.2 Input and output files

The specification of input files is detailed in the decryption of the `init` command in Section 3.3.1 – `DATA_EN` is the source English text (input of an MT system), `DATA_CS` is its Czech machine translation (output of the MT system), and `REFERENCE_CS` is a reference Czech translation of the source English text by a human translator.

The default values correspond to the sample one-sentence input used for the `test` command:

```
DATA_CS=data/csw.txt DATA_EN=data/en.txt REFERENCE_CS=data/cs.txt
```

You can also use the larger sample input which is bundled with Depfix:

```
DATA_CS=data/wmt10_bojar-2012.cs DATA_EN=data/wmt10_src.en
  REFERENCE_CS=data/wmt10_ref.cs
```

The format of the input files must be plain text in UTF-8 encoding, one sentence per line. All of the input files must have the same number of lines. Currently, Depfix requires you to provide a reference translation file; if you do not have one, please set `REFERENCE_CS` to any other file (which has the correct number of lines); we recommend setting to the same value as `DATA_CS`.

The input files get copied into the experiment directory under the following filenames: `data_cs.txt`, `data_en.txt`, and `ref_cs.txt`. The output of the Depfix pipeline is stored into a text file in the experiment directory, specified by `OUTPUT_TXT` (the default is `output.txt`); the log which contains information on fixes that were applied is stored in `fixlog.txt`.

The `compare*` commands (see Section 3.3.2) operate on the inputs and outputs of Depfix by default, but can be parametrized to behave differently by setting the following variables:

ORI original Czech translation (defaults to `data_cs.txt`)

NEW Depfix output (defaults to `OUTPUT_TXT`)

REF reference Czech translation (defaults to `REF_CS_TXT`, which defaults to `ref_cs.txt`)

EN source English sentences (defaults to `data_en.txt`)

The output of automatic evaluation is stored into a text file specified by `AUTOEVAL_OUT`; the default is `autoeval.out`.

3.4.3 Depfix operation

DIRLABEL is the label to use for the new directory created by `init` (or `default`, or `clone`).

TREEX is the command to invoke Treex (defaults to `treex`); you may add some Treex options in special cases

TREEXP is the command to invoke Treex in parallel, i.e. to perform commands that can be run independently on individual Treex files (generally, all commands except those that work with text files – these use `TREEX` instead of `TREEXP`). It defaults to `treex`, which means that Treex is never run

in parallel by default. If you have an SGE cluster, you may use settings such as the following to run Treex in parallel by default (it is reasonable to set these in the `settings.mak` file, as described in Section 3.4.1):

```
JOBS=10
MEM=30G
WORKDIR=$$(mktemp -d --tmpdir=$(DIRNAME))
TREEXP=treex -p --survive --jobs=$(JOBS) --mem=$(MEM) \
  --workdir=$(WORKDIR)
```

3.4.4 Scenarios

Most of the settings specify scenarios to use for various commands, as described in the previous section. The scenarios themselves are contained in the `depfix/scenarios` subdirectory. In many cases, there are alternative scenarios available that you may use in some situations to modify the way Depfix operates, such as to use a different parser or to omit some processing steps; the name of an alternative scenario typically has an identical prefix to that of the default scenario.

Also, there is the `empty.scen` scenario, which does not do anything, so you may use it to bypass some scenario completely (although this will fail in some cases, as many of the steps are vital for the operation of the subsequent steps). For example, to skip running Czech named entity recognizer, you may use the following setting:

```
NER_CS_SCEN=empty.scen
```

We list here the settings, their default values, and some of the alternatives in cases where we find it useful. The names of the settings are rather informative, and they are given in the order in which they are used in the default Depfix pipeline; to see exactly where they are used, see Section 3.3.

PROJECT_TOKENIZATION_SCEN Default value: `project_tokenization.scen`

EN_ANALYSIS_1_SCEN Default value: `en_analysis_1.scen`. Alternative value: `en_analysis_1_old.scen` to use the Morče tagger

CS_ANALYSIS_1_SCEN Default value: `cs_analysis_1_morphodita.scen`. Alternative values: `cs_analysis_1_featurama_fme.scen` to use the Featurama tagger, or `cs_analysis_1_morce.scen` to use the Morče tagger

RUN_GIZA_SCEN Default value: `run_mgiza.scen`. Alternative value: `run_mgiza_ufal.scen` to be used on machines of our institute (ÚFAL); the important difference is `tmp_dir=/COMP.TMP` instead of `tmp_dir=/tmp` – if your `/tmp` directory is on a network drive, you should use a similar modification so that a local directory is used instead

ADD_MISSING_LINKS_SCEN Default value: `add_missing_links.scen`

NER_EN_SCEN Default value: `ner_en.scen`

EN_ANALYSIS_2_SCEN Default value: `en_analysis_2.scen`

CS_ANALYSIS_2_SCEN Default value: `cs_analysis_2_msta.scen`. Alternative value: `cs_analysis_2_boost_model_025.scen` to use the MSTperl parser adapted for parsing SMT outputs (see Section 3.2)

NER_CS_SCEN Default value: `ner_cs.scen`

FIX_PREPARE_SCEN Default value: `fix_prepare.scen`

NER_CS_SCEN Default value: `ner_cs.scen`

BEFORE_FIX Default value: *empty*. May be used to run a scenario before running `fix.scen` in the `fix_run` command

FIX_SCEN Default value: `fix.scen`

A2T_CS_SCEN Default value: `a2t_cs.scen`

A2T_EN_SCEN Default value: `a2t_en.scen`

TFIX_PREPARE_SCEN Default value: `tfix_prepare.scen`

TFIX_SCEN Default value: `tfix_rules.scen`. Alternative values: `tfix_cut_ChgCase2.scen` to also run statistical fixes (see Section 3.2); `tfix_stat.scen` to run only the statistical fixes, i.e. not running the rules based ones.

REFIX_SCEN Default value: `$(FIX_SCEN)`

WRITE_SENTENCES_SCEN Default value: `write_sentences.scen`. Alternative value: `write_sentences_no_detok.scen` not to perform detokenization, i.e. to write out individual tokens separated by spaces.

3.4.5 Evaluation

The following variables are used for manual evaluation (see Section 3.3.3):

ANNOT_LINES The number of sentences to be selected for evaluation (including unchanged ones, so the final number of sentences is likely to be lower). Default value: `3003`

ANNOT_NUM Number of annotation files to generate – usually should be a multiple of the annotators you have, as each of the generated files is to be given to one of your annotators. Default value: `20`

MAN_PREFIX Prefix to use for the directory created for manual evaluation files. Default value: `depfix_maneval`

The following variables are used for bootstrap significance tests in automatic evaluation (see Section 3.3.4):

SAMPLES The number of repetitions of the experiment. Default value: `1000`

ALPHA The significance level. Default value: `0.05`

4 Delving deeper

If you want to modify Depfix operation more than what is allowed by changing the settings of the variables, you need to go into the source codes – the `Makefile`, the scenarios, and/or the Treex blocks.

One possibility you have is modifying the scenarios used by Depfix; we suggest that you do that by copying an existing scenario, modifying it, and then calling Depfix with the corresponding value set to the filename of such new scenario. All the scenarios are plaintext files that reside in the `depfix/scenarios` subfolder; do put even your new scenarios in that folder for smooth operarion. A scenario typically sets several global arguments, and then invokes a pipeline of Depfix blocks, sometimes with arguments set. See the Treex manual for more information about Treex scenarios.

You may also need to edit the `Makefile` to suit your modified scenarios; see a `make` manual if you need to.

The key scenario which invokes most of the fixes is `fix.scen`. The fixes are Treex blocks, and thus can be found in subdirectories of `treex/lib/Treex/Block`; most of them are in the `treex/lib/Treex/Block/A2A/CS` directory (but not everything in this directory are Depfix blocks; Depfix block names typically start with the word `Fix`).

There are also several Depfix-specific Treex tools used; you will find them in `treex/lib/Treex/Tool/Depfix`. These are mainly wrappers for existing tools to enable easy operation from within Depfix.

Many Treex blocks and tools contain documentation in POD (Plain Old Documentation) format and can be viewed using the `perldoc` tool,[10] e.g.:

```
perldoc Treex::Block::A2A::CS::FixGenitive
```

Moreover, the source codes are usually commented.

5 Support

If you encounter any issues you are unable to solve yourself, or if you have any questions regarding Depfix, feel free to contact the author at `rosa@ufal.mff.cuni.cz`. Issues and questions that concern Treex (and not specifically Depfix) should preferably be directed to `treex@ufal.mff.cuni.cz`.

Acknowledgements

Depfix is supported by the grants FP7-ICT-2013-10-610516 (QTLeap), GAUK 1572314, and SVV 260 104. This work has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013).

References

[1] David Mareček, Rudolf Rosa, Petra Galuščáková, and Ondřej Bojar. Two-step translation with grammatical post-processing. In Chris Callison-Burch,

[10] `http://perldoc.perl.org/perldoc.html`

Philipp Koehn, Christof Monz, and Omar Zaidan, editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 426–432, Edinburgh, UK, 2011. University of Edinburgh, Association for Computational Linguistics.

[2] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, 2005.

[3] Václav Novák and Zdeněk Žabokrtský. Feature engineering in maximum spanning tree dependency parser. In Václav Matoušek and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, Lecture Notes in Computer Science, pages 92–98, Pilsen, Czech Republic, 2007. Springer Science+Business Media Deutschland GmbH.

[4] Rudolf Rosa. Automatic post-editing of phrase-based machine translation outputs. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2013.

[5] Rudolf Rosa. Depfix, a tool for automatic rule-based post-editing of SMT. *The Prague Bulletin of Mathematical Linguistics*, 102:47–56, 2014.

[6] Rudolf Rosa. MSTperl parser, 2014. http://hdl.handle.net/11858/00-097C-0000-0023-7AEB-4.

[7] Rudolf Rosa, Ondřej Dušek, David Mareček, and Martin Popel. Using parallel features in parsing of machine-translated sentences for correction of grammatical errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea, 2012. Association for Computational Linguistics.

[8] Rudolf Rosa, David Mareček, and Ondřej Dušek. DEPFIX: A system for automatic correction of Czech MT outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, 2012. Association for Computational Linguistics.

[9] Rudolf Rosa, David Mareček, and Aleš Tamchyna. Deepfix: Statistical post-editing of statistical machine translation using deep syntactic analysis. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 172–179, Sofija, Bulgaria, 2013. Bălgarska akademija na naukite, Association for Computational Linguistics.

The ÚFAL/CKL Technical Report Series

ÚFAL

ÚFAL (Ústav formální a aplikované lingvistiky; http://ufal.mff.cuni.cz) is the Institute of Formal and Applied linguistics, at the Faculty of Mathematics and Physics of Charles University, Prague, Czech Republic. The Institute was established in 1990 after the political changes as a continuation of the research work and teaching carried out by the former Laboratory of Algebraic Linguistics since the early 60s at the Faculty of Philosophy and later the Faculty of Mathematics and Physics. Together with the "sister" Institute of Theoretical and Computational Linguistics (Faculty of Arts) we aim at the development of teaching programs and research in the domain of theoretical and computational linguistics at the respective Faculties, collaborating closely with other departments such as the Institute of the Czech National Corpus at the Faculty of Philosophy and the Department of Computer Science at the Faculty of Mathematics and Physics.

CKL

As of 1 June 2000 the Center for Computational Linguistics (Centrum komputační lingvistiky; http://ckl.mff.cuni.cz) was established as one of the centers of excellence within the governmental program for support of research in the Czech Republic. The center is attached to the Faculty of Mathematics and Physics of Charles University in Prague.

Technical Reports

The ÚFAL/CKL technical report series has been established with the aim of disseminate topical results of research currently pursued by members, cooperators, or visitors of the Institute. The technical reports published in this Series are results of the research carried out in the research projects supported by the Grant Agency of the Czech Republic, GAČR 405/96/K214 ("Komplexní program"), GAČR 405/96/0198 (Treebank project), grant of the Ministry of Education of the Czech Republic VS 96151, and project of the Ministry of Education of the Czech Republic LN00A063 (Center for Computational Linguistics). Since November 1996, the following reports have been published.

ÚFAL TR-1996-01 Eva Hajičová, *The Past and Present of Computational Linguistics at Charles University*
Jan Hajič and Barbora Hladká, *Probabilistic and Rule-Based Tagging of an Inflective Language – A Comparison*

ÚFAL TR-1997-02 Vladislav Kuboň, Tomáš Holan and Martin Plátek, *A Grammar-Checker for Czech*

ÚFAL TR-1997-03 Alla Bémová at al., Anotace na analytické rovině, *Návod pro anotátory (in Czech)*

ÚFAL TR-1997-04 Jan Hajič and Barbora Hladká, *Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structural Tagset*

ÚFAL TR-1998-05 Geert-Jan M. Kruijff, *Basic Dependency-Based Logical Grammar*

ÚFAL TR-1999-06 Vladislav Kuboň, *A Robust Parser for Czech*

ÚFAL TR-1999-07 Eva Hajičová, Jarmila Panevová and Petr Sgall, *Manuál pro tektogramatické značkování (in Czech)*

ÚFAL TR-2000-08 Tomáš Holan, Vladislav Kuboň, Karel Oliva, Martin Plátek, *On Complexity of Word Order*

ÚFAL/CKL TR-2000-09 Eva Hajičová, Jarmila Panevová and Petr Sgall, *A Manual for Tectogrammatical Tagging of the Prague Dependency Treebank*

ÚFAL/CKL TR-2001-10 Zdeněk Žabokrtský, *Automatic Functor Assignment in the Prague Dependency Treebank*

ÚFAL/CKL TR-2001-11 Markéta Straňáková, *Homonymie předložkových skupin v češtině a možnost jejich automatického zpracování*

ÚFAL/CKL TR-2001-12 Eva Hajičová, Jarmila Panevová and Petr Sgall, *Manuál pro tektogramatické značkování (III. verze)*

ÚFAL/CKL TR-2002-13 Pavel Pecina and Martin Holub, *Sémanticky signifikantní kolokace*

ÚFAL/CKL TR-2002-14 Jiří Hana, Hana Hanová, *Manual for Morphological Annotation*

ÚFAL/CKL TR-2002-15 Markéta Lopatková, Zdeněk Žabokrtský, Karolína Skwarská and Vendula Benešová, *Tektogramaticky anotovaný valenční slovník českých sloves*

ÚFAL/CKL TR-2002-16 Radu Gramatovici and Martin Plátek, *D-trivial Dependency Grammars with Global Word-Order Restrictions*

ÚFAL/CKL TR-2003-17 Pavel Květoň, *Language for Grammatical Rules*

ÚFAL/CKL TR-2003-18 Markéta Lopatková, Zdeněk Žabokrtský, Karolina Skwarska, Václava Benešová, *Valency Lexicon of Czech Verbs VALLEX 1.0*

ÚFAL/CKL TR-2003-19 Lucie Kučová, Veronika Kolářová, Zdeněk Žabokrtský, Petr Pajas, Oliver Čulo, *Anotování koreference v Pražském závislostním korpusu*

ÚFAL/CKL TR-2003-20 Kateřina Veselá, Jiří Havelka, *Anotování aktuálního členění věty v Pražském závislostním korpusu*

ÚFAL/CKL TR-2004-21 Silvie Cinková, *Manuál pro tektogramatickou anotaci angličtiny*

ÚFAL/CKL TR-2004-22 Daniel Zeman, *Neprojektivity v Pražském závislostním korpusu (PDT)*

ÚFAL/CKL TR-2004-23 Jan Hajič a kol., *Anotace na analytické rovině, návod pro anotátory*

ÚFAL/CKL TR-2004-24 Jan Hajič, Zdeňka Urešová, Alevtina Bémová, Marie Kaplanová, *Anotace na tektogramatické rovině (úroveň 3)*

ÚFAL/CKL TR-2004-25 Jan Hajič, Zdeňka Urešová, Alevtina Bémová, Marie Kaplanová, *The Prague Dependency Treebank, Annotation on tectogrammatical level*

ÚFAL/CKL TR-2004-26 Martin Holub, Jiří Diviš, Jan Pávek, Pavel Pecina, Jiří Semecký, *Topics of Texts. Annotation, Automatic Searching and Indexing*

ÚFAL/CKL TR-2005-27 Jiří Hana, Daniel Zeman, *Manual for Morphological Annotation (Revision for PDT 2.0)*

ÚFAL/CKL TR-2005-28 Marie Mikulová a kol., *Pražský závislostní korpus (The Prague Dependency Treebank) Anotace na tektogramatické rovině (úroveň 3)*

ÚFAL/CKL TR-2005-29 Petr Pajas, Jan Štěpánek, *A Generic XML-Based Format for Structured Linguistic Annotation and Its application to the Prague Dependency Treebank 2.0*

ÚFAL/CKL TR-2006-30 Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Annotation on the tectogrammatical level in the Prague Dependency Treebank (Annotation manual)*

ÚFAL/CKL TR-2006-31 Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Petr Sgall, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Anotace na tektogramatické rovině Pražského závislostního korpusu (Referenční příručka)*

ÚFAL/CKL TR-2006-32 Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Petr Sgall,Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Annotation on the tectogrammatical level in the Prague Dependency Treebank (Reference book)*

ÚFAL/CKL TR-2006-33 Jan Hajič, Marie Mikulová, Martina Otradovcová, Petr Pajas, Petr Podveský, Zdeňka Urešová, *Pražský závislostní korpus mluvené češtiny. Rekonstrukce standardizovaného textu z mluvené řeči*

ÚFAL/CKL TR-2006-34 Markéta Lopatková, Zdeněk Žabokrtský, Václava Benešová (in cooperation with Karolína Skwarska, Klára Hrstková, Michaela Nová, Eduard Bejček, Miroslav Tichý) *Valency Lexicon of Czech Verbs. VALLEX 2.0*

ÚFAL/CKL TR-2006-35 Silvie Cinková, Jan Hajič, Marie Mikulová, Lucie Mladová, Anja Nedolužko, Petr Pajas, Jarmila Panevová, Jiří Semecký, Jana Šindlerová, Josef Toman, Zdeňka Urešová, Zdeněk Žabokrtský, *Annotation of English on the tectogrammatical level*

ÚFAL/CKL TR-2007-36 Magda Ševčíková, Zdeněk Žabokrtský, Oldřich Krůza, *Zpracování pojmenovaných entit v českých textech*

ÚFAL/CKL TR-2008-37 Silvie Cinková, Marie Mikulová, *Spontaneous speech reconstruction for the syntactic and semantic analysis of the NAP corpus*

ÚFAL/CKL TR-2008-38 Marie Mikulová, *Rekonstrukce standardizovaného textu z mluvené řeči v Pražském závislostním korpusu mluvené češtiny. Manuál pro anotátory*

ÚFAL/CKL TR-2008-39 Zdeněk Žabokrtský, Ondřej Bojar, *TectoMT, Developer's Guide*

ÚFAL/CKL TR-2008-40 Lucie Mladová, *Diskurzní vztahy v češtině a jejich zachycení v Pražském závislostním korpusu 2.0*

ÚFAL/CKL TR-2009-41 Marie Mikulová, *Pokyny k překladu určené překladatelům, revizorům a korektorům textů z Wall Street Journal pro projekt PCEDT*

ÚFAL/CKL TR-2011-42 Loganathan Ramasamy, Zdeněk Žabokrtský, *Tamil Dependency Treebank (TamilTB) – 0.1 Annotation Manual*

ÚFAL/CKL TR-2011-43 Nguy Giang Linh, Michal Novák, Anna Nedoluzhko, *Coreference Resolution in the Prague Dependency Treebank*

ÚFAL/CKL TR-2011-44 Anna Nedoluzhko, Jiří Mírovský, *Annotating Extended Textual Coreference and Bridging Relations in the Prague Dependency Treebank*

ÚFAL/CKL TR-2011-45 David Mareček, Zdeněk Žabokrtský, *Unsupervised Dependency Parsing*

ÚFAL/CKL TR-2011-46 Martin Majliš, Zdeněk Žabokrtský, *W2C – Large Multilingual Corpus*

ÚFAL TR-2012-47 Lucie Poláková, Pavlína Jínová, Šárka Zikánová, Zuzanna Bedřichová, Jiří Mírovský, Magdaléna Rysová, Jana Zdeňková, Veronika Pavlíková, Eva Hajičová, *Manual for annotation of discourse relations in the Prague Dependency Treebank*

ÚFAL TR-2012-48 Nathan Green, Zdeněk Žabokrtský, *Ensemble Parsing and its Effect on Machine Translation*

ÚFAL TR-2013-49 David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Daniel Zemana, Zdeněk Žabokrtský, Jan Hajič *Cross-language Study on Influence of Coordination Style on Dependency Parsing Performance*

ÚFAL TR-2013-50 Jan Berka, Ondřej Bojar, Mark Fishel, Maja Popović, Daniel Zeman, *Tools for Machine Translation Quality Inspection*

ÚFAL TR-2013-51 Marie Mikulová, *Anotace na tektogramatické rovině. Dodatky k anotátorské příručce (s ohledem na anotování PDTSC a PCEDT)*

ÚFAL TR-2013-52 Marie Mikulová, *Annotation on the tectogrammatical level. Additions to annotation manual (with respect to PDTSC and PCEDT)*

ÚFAL TR-2013-53 Marie Mikulová, Eduard Bejček, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Pavel Straňák, Magda Ševčíková, Zdeněk Žabokrtský, *Úpravy a doplňky Pražského závislostního korpusu (Od PDT 2.0 k PDT 3.0)*

ÚFAL TR-2013-54 Marie Mikulová, Eduard Bejček, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Pavel Straňák, Magda Ševčíková, Zdeněk Žabokrtský, *From PDT 2.0 to PDT 3.0 (Modifications and Complements)*

ÚFAL TR-2014-55 Rudolf Rosa, *Depfix Manual*