Install Dutch nlp modules on Lisa

Paul Huygen <paul.huygen@huygen.nl>

30th January 2015
12:36 h.

Abstract

This is a description and documentation of the installation of the current NLP modules on Lisa, so that they can be used in pipelines.

Contents

1 Introduction

This document describes the current set-up of pipeline that annotates dutch texts in order to extract knowledge. The pipeline has been set up by the Computational Lexicology an Terminology Lab (CLTL [1]) as part of the newsreader [2].

Apart from describing the pipeline set-up, the document actually constructs the pipeline. The described version has been made with an aim to run it on a specific supercomputer (Lisa, Surfsara, Amsterdam [3]), but it can probably be implemented on other unix-like systems without problems.

The installation has been parameterized. The locations and names that you read (and that will be used to build the pipeline) have been read from variables in file `inst.m4` in the nuweb directory.

1.1 List of the modules to be installed

Table 1 lists the modules in the pipeline. The column *source* indicates the origin of the module. The modules are obtained in one of the following ways:

1. If possible, the module is directly obtained from an open-source repository like Github.
2. Some modules have not been officially published in a repository. These modules have been packed in a tar-ball that can be obtained by the author. This is indicated as TAR.

1. http://wordpress.let.vupr.nl
2. http://www.newsreader-project.eu
3. https://surfsara.nl/systems/lisa

Module	Section	Source	Script	Details
Tokenizer	3.6.1	Github	`tok`	
morphosyntactic parser	3.6.2	Github	`mor`	
NERC	3.6.5	Github	nerc	
WSD	3.6.6	Github	wsd	
Onto-tagger	3.6.9	snapshot	onto	
Heideltime	3.6.11	Github	heideltime	
SRL	3.6.12	Github	srl	
NED	3.6.8	Github	ned	
Nom. coref	3.6.4	Github	nomcoref	
Ev. coref	3.6.13	snapshot	evcoref	
Framenet sem. role label.	3.6.10	snapshot	fsrl	

Table 1: *List of the modules to be installed. Column description:* ***directory:*** *Name of the subdirectory below subdirectory* `modules` *in which it is installed;* ***Source:*** *From where the module has been obtained;* ***script:*** *Script to be included in a pipeline.*

The modules themselves use other utilities like dependency-taggers and POS taggers. These utilities are listed in table 2.

Module	Section	Source
KafNafParserPy	2.3.2	Github
Alpino	3.5.1	RUG
Ticcutils	3.5.3	ILK
Timbl	3.5.3	ILK
Treetagger	3.5.2	Uni. München

Table 2: *List of the modules to be installed. Column description:* ***directory:*** *Name of the subdirectory below* `mod` *in which it is installed;* ***Source:*** *From where the module has been obtained;* ***script:*** *Script to be included in a pipeline.*

1.2 File-structure of the pipeline

The files that make up the pipeline are organised in set of directories:

nuweb: This directory contains this document and everything to create the pipeline from the open sources of the modules.

modules: Contains the program code of each module in a subdirectory. Furthermore, it contains a subdirectory `python` for python software-modules, subdirectory `jars` for jar files and subdirectory /usrlocal/ for binaries and libs that are used by modules.

bin: Contains for each of the modules a script that reads NAF input, passes it to the module in the `modules` directory and produces the output on standard out. Furthermore, the subdirectory contains the script `install-modules` that performs the installation, and a script `test` that shows that the pipeline works in a trivial case.

nuweb: Contains this document, the nuweb source that creates the documents and the sources and a Makefile to perform the actions.

⟨ *directories to create* 3a ⟩ ≡
```
../modules ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *directories to create* 3b ⟩ ≡
```
../bin ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *directories to create* 4a ⟩ ≡
```
../modules/usrlocal ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *directories to create* 4b ⟩ ≡
```
../modules/usrlocal/bin ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *directories to create* 4c ⟩ ≡
```
../modules/usrlocal/lib ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *directories to create* 4d ⟩ ≡
```
../modules/python ../env/java/jars ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

Make binaries findable:

⟨ *set local bin directory* 4e ⟩ ≡
```
export PATH=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/usrlocal/bin:$PATH
◇
```
Fragment referenced in 27e.

2 Java and Python environment

To be independent from the software environment of the host computer and to perform reproducible processing, the pipeline features its own Java and Python environment. The costs of this feature are that the pipeline takes more disk-space by reproducing infra-structure that is already present in the system and that installation takes more time.

The following file sets up the programming environment in scripts.

"../bin/progenv" 4f≡
```
PIPEROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
PIPEBIN=$PIPEROOT/bin
PIPEMODD=$PIPEROOT/modules
⟨ set up java environment in scripts 5e, ... ⟩
⟨ activate the python environment 7c, ... ⟩
◇
```

⟨ *set up programming environment* 4g ⟩ ≡
```
source /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/bin/progenv
◇
```
Fragment referenced in 17b, 18a, 19g, 20d, 22a, 24b, 25b, 28d.

2.1 Java

To install Java, download `server-jre-7u72-linux-x64.tar.gz` from http://www.oracle.com/technetwork/java/javase/downloads/server-jre7-downloads-1931105.html. Find it in the root directory and unpack it in a subdirectory of `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env`.

⟨ *directories to create* 5a ⟩ ≡
```
../env/java ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *check this first* 5b ⟩ ≡
```
if
  [ ! -e /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/server-jre-
7u72-linux-x64.tar.gz ]
then
  echo "Cannot find  /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/server-jre-7u72-linux-x64.tar.gz"
  exit 4
fi
◇
```
Fragment referenced in 10b.

⟨ *set up java* 5c ⟩ ≡
```
⟨ unpack the java tarball 5d ⟩
◇
```
Fragment referenced in 10b.

⟨ *unpack the java tarball* 5d ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java
tar -xzf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/server-jre-
7u72-linux-x64.tar.gz
◇
```
Fragment referenced in 5c.

⟨ *set up java environment in scripts* 5e ⟩ ≡
```
export JAVA_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 5e, 6a.
Fragment referenced in 4f, 10b.
Defines: `JAVA_HOME` Never used.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 5f ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *set up java environment in scripts* 6a ⟩ ≡

```
export JARDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/java/jars
◇
```

Fragment defined by 5e, 6a.
Fragment referenced in 4f, 10b.

2.2 Maven

⟨ *directories to create* 6b ⟩ ≡

```
/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/apache-maven-
3.0.5 ◇
```

Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

⟨ *install maven* 6c ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env
wget http://apache.rediris.es/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-
bin.tar.gz
tar -xzf apache-maven-3.0.5-bin.tar.gz
rm apache-maven-3.0.5-bin.tar.gz
◇
```

Fragment defined by 6cd.
Fragment referenced in 10b.

⟨ *install maven* 6d ⟩ ≡

```
export MAVEN_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```

Fragment defined by 6cd.
Fragment referenced in 10b.

⟨ *remove maven* 6e ⟩ ≡

```
rm -rf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/apache-
maven-3.0.5
◇
```

Fragment referenced in 11b.

2.3 Python

⟨ *set up python* 6f ⟩ ≡

```
⟨ create a virtual environment for Python 7a ⟩
⟨ activate the python environment 7c, ... ⟩
⟨ install kafnafparserpy 8a ⟩
⟨ install python packages 8f ⟩
◇
```

Fragment referenced in 10b.

2.3.1 Virtual environment

Create a virtual environment.

⟨ *create a virtual environment for Python* 7a ⟩ ≡
⟨ *test whether virtualenv is present on the host* 7b ⟩
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env
virtualenv venv
```
◇

Fragment referenced in 6f.
Uses: `virtualenv` 7b.

⟨ *test whether virtualenv is present on the host* 7b ⟩ ≡
```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
```
◇

Fragment referenced in 7a.
Defines: `virtualenv` 7a.

⟨ *activate the python environment* 7c ⟩ ≡
```
source /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/venv/bin/activate
```
◇

Fragment defined by 7ce.
Fragment referenced in 4f, 6f, 26f, 27e.
Defines: `activate` Never used.

Subdirectory `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python` will contain general Python packages like KafnafParserPy.

⟨ *directories to create* 7d ⟩ ≡
```
/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python
```
◇

Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

Activation of Python include pointing to the place where Python packages are:

⟨ *activate the python environment* 7e ⟩ ≡
```
export PYTHONPATH=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/python:$PYTHONPATH
```
◇

Fragment defined by 7ce.
Fragment referenced in 4f, 6f, 26f, 27e.
Defines: `PYTHONPATH` Never used.

2.3.2 KafNafParserPy

A cornerstone Pythonmodule for the pipeline is KafNafParserPy. It is a feature of this module that it cannot be installed with PIP, but that you can put it somewhere and then put the somewhere

in your PYTHONPATH.

⟨ *install kafnafparserpy* 8a ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python
DIRN=KafNafParserPy
⟨ move module (8b $DIRN ) 8g ⟩
git clone https://github.com/cltl/KafNafParserPy.git
if
  [ $? -gt 0 ]
then
  ⟨ logmess (8c Cannot install current $DIRN version ) 30a ⟩
  ⟨ re-instate old module (8d $DIRN ) 9b ⟩
else
  ⟨ remove old module (8e $DIRN ) 9a ⟩
fi
◇
```
Fragment referenced in 6f.

2.3.3 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* 8f ⟩ ≡
```
pip install lxml
pip install pyyaml
◇
```
Fragment referenced in 6f.
Defines: `lxml` Never used, `pyyaml` Never used.

3 Installation

This section describes how the modules are obtained from their (open-)source and installed.

3.1 Installing vs. updating

When the install-script installs something that has already been installed, it moves the installed module to a temporary location and then tries to install the module from its source. If that is successfull it removes the vormer version of the module, otherwise it moves the old version back.

The following macro's can be used to move or remove modules, provided they are called when the modules directory is the default directory.

⟨ *move module* 8g ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
◇
```
Fragment referenced in 8a, 9d, 12a, 30b.

⟨ *remove old module* 9a ⟩ ≡
```
rm -rf old.@1
```
◇
Fragment referenced in 8a, 9d, 12a, 30b.

⟨ *re-instate old module* 9b ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
⟨ logmess (9c $MESS ) 30a ⟩

```
◇
Fragment referenced in 8a, 9d, 12a, 30b.

3.2 Installation from Github

The following macro can be used to install a module from github. It needs as parameters:

1. Name of the module.
2. Name of the root directory.
3. Github URL to clone from.

⟨ *install from github* 9d ⟩ ≡
```
MODNAM=@1
DIRN=@2
GITU=@3
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (9e $DIRN ) 8g ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (9f Cannot install current $MODNAM version ) 30a ⟩
  ⟨ re-instate old module (9g $DIRN ) 9b ⟩
else
  ⟨ remove old module (9h $DIRN ) 9a ⟩
fi

```
◇
Fragment referenced in 17d, 19c, 21a, 23a, 26a, 27a.

3.3 Installation from the snapshot

For some modules a public repository is not available or not known. They must be installed from a tarball with snapshots that can be obtained from the author. Let us first check whether we have the snapshot and complain if we don't. We expect the file `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/nl-pipeline_snapshots_20150127.tgz`.

⟨ *unpack snapshots or die* 10a ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
if
  [ -e nl-pipeline_snapshots_20150127.tgz ]
then
  tar -zxf nl-pipeline_snapshots_20150127.tgz
fi
if
  [ ! -e snapshots ]
then
  echo "No module snapshots"
  exit 1
fi
◇
```

Fragment referenced in 10b.

3.4 The installation script

The installation is performed by script `install-modules`

`"../bin/install-modules"` 10b≡

```
#!/bin/bash
echo Set up environment
⟨ variables of install-modules 29d ⟩
⟨ check this first 5b ⟩
⟨ unpack snapshots or die 10a ⟩
echo ... Java
⟨ set up java 5c ⟩
⟨ set up java environment in scripts 5e, ... ⟩
⟨ install maven 6c, ... ⟩
echo ... Python
⟨ set up python 6f ⟩
echo ... Alpino
⟨ install Alpino 12a ⟩
echo ... Spotlight
⟨ install the spotlight server 16a, ... ⟩
echo ... Treetagger
⟨ install the treetagger utility 12g, ... ⟩
echo ... Ticcutils and Timbl
⟨ install the ticcutils utility 14c ⟩
⟨ install the timbl utility 14d ⟩
◇
```

File defined by 10b, 11ab.

```
"../bin/install-modules" 11a≡
        echo Install modules
        echo ... Tokenizer
        ⟨ install the tokenizer 17a ⟩
        echo ... Morphosyntactic parser
        ⟨ install the morphosyntactic parser 17d ⟩
        echo ... NERC
        ⟨ install the NERC module 20a ⟩
        echo ... Coreference base
        ⟨ install coreference-base 19c ⟩
        echo ... WSD
        ⟨ install the WSD module 21a ⟩
        echo ... Ontotagger
        ⟨ install the onto module 24a ⟩
        echo ... Heideltime
        ⟨ install the heideltime module 26a ⟩
        echo ... SRL
        ⟨ install the srl module 27a ⟩
        echo ... NED
        ⟨ install the NED module 23a ⟩
        echo ... Event-coreference
        ⟨ install the event-coreference module 28c ⟩
        echo Final
        ◇
```

File defined by 10b, 11ab.

```
"../bin/install-modules" 11b≡
        ⟨ remove maven 6e ⟩
        ◇
```

File defined by 10b, 11ab.

```
⟨ make scripts executable 11c ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.5 Install utilities and resources

3.5.1 Alpino

Install Alpino from the website of Gertjan van Noort.

Module

⟨ *install Alpino* 12a ⟩ ≡

```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (12b Alpino ) 8g ⟩
wget http://www.let.rug.nl/vannoord/alp/Alpino/binary/versions/Alpino-x86_64-linux-
glibc2.5-20548-sicstus.tar.gz
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz
  SUCCES=$?
  rm -rf Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (12c Installed Alpino ) 30a ⟩
  ⟨ remove old module (12d Alpino ) 9a ⟩
else
  ⟨ re-instate old module (12e Alpino ) 9b ⟩
fi
◇
```

Fragment referenced in 10b.

Currently, alpino is not used as a pipeline-module on its own, but it is included in other pipeline-modules. Modules that use Alpino should set the following variables:

⟨ *set alpinohome* 12f ⟩ ≡

```
export ALPINO_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/Alpino
◇
```

Fragment referenced in 18a.
Defines: `ALPINO_HOME` Never used.

3.5.2 Treetagger

Installation of Treetagger goes as follows (See Treetagger's homepage:

1. Download and unpack the treetagger tarball. This generates the subdirectories `bin`, `cmd` and `doc`
2. Download and unpack the tagger-scripts tarball

The location where treetager comes from and the location where it is going to reside:

⟨ *install the treetagger utility* 12g ⟩ ≡

```
TREETAGDIR=treetagger
TREETAG_BASIS_URL=http://www.cis.uni-muenchen.de/%7Eschmid/tools/TreeTagger/data/
TREETAGURL=http://www.cis.uni-muenchen.de/%7Eschmid/tools/TreeTagger/data/
◇
```

Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 13a ⟩ ≡
```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
◇
```
Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

Parametersets:

⟨ *install the treetagger utility* 13b ⟩ ≡
```
DUTCHPARS_UTF_GZ=dutch-par-linux-3.2-utf8.bin.gz
DUTCH_TAGSET=dutch-tagset.txt
DUTCHPARS_2_GZ=dutch2-par-linux-3.2-utf8.bin.gz
◇
```
Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

Download everything in the target directory:

⟨ *install the treetagger utility* 13c ⟩ ≡
```
mkdir -p /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR
wget $TREETAGURL/$TREETAGSRC
wget $TREETAGURL/$TREETAGSCRIPTS
wget $TREETAGURL/$TREETAG_INSTALLSCRIPT
wget $TREETAGURL/$DUTCHPARS_UTF_GZ
wget $TREETAGURL/$DUTCH_TAGSET
wget $TREETAGURL/$DUTCHPARS_2_GZ
◇
```
Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

Run the install-script:

⟨ *install the treetagger utility* 13d ⟩ ≡
```
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```
Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

Make the treetagger utilities available for everbody.

⟨ *install the treetagger utility* 14a ⟩ ≡

```
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR/bin
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR/cmd
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR/doc
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/$TREETAGDIR/lib
./$TREETAG_INSTALLSCRIPT
◇
```

Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

Remove the tarballs:

⟨ *install the treetagger utility* 14b ⟩ ≡

```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```

Fragment defined by 12g, 13abcd, 14ab.
Fragment referenced in 10b.

3.5.3 Timbl and ticcutils

Timbl and ticcutils are installed from their source-tarballs. The installation is not (yet?) completely reproducibe because it uses the c-compiler that happens to be available on the host. Installation involves:

1. Download the tarball in a temporary directory.
2. Unpack the tarball.
3. cd to the unpacked directory and perform `./configure`, `make` and `make install`. Note the argument that causes the files to be installed in the `usrlocal` subdirectory of the modules directory.

⟨ *install the ticcutils utility* 14c ⟩ ≡

```
URL=http://software.ticc.uvt.nl/ticcutils-0.7.tar.gz
TARB=ticcutils-0.7.tar.gz
DIR=ticcutils-0.7
⟨ unpack ticcutils or timbl 15a ⟩
◇
```

Fragment referenced in 10b.

⟨ *install the timbl utility* 14d ⟩ ≡

```
URL=http://software.ticc.uvt.nl/timbl-6.4.6.tar.gz
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
⟨ unpack ticcutils or timbl 15a ⟩
◇
```

Fragment referenced in 10b.

⟨ *unpack ticcutils or timbl* 15a ⟩ ≡

```
SUCCES=0
ticbeldir=‘mktemp -t -d tickbel.XXXXXX‘
cd $ticbeldir
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  cd $DIR
  ./configure --prefix=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/usrlocal
  make
  make install
fi
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
rm -rf $ticbeldir
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (15b Installed $DIR ) 30a ⟩
else
  ⟨ logmess (15c NOT installed $DIR ) 30a ⟩
fi
◇
```

Fragment referenced in 14cd.

3.5.4 Spotlight

Install spotlight in the way that Itziar Aldabe (mailto:itziar.aldabe@ehu.es) described:

> The NED module works for English, Spanish, Dutch and Italian. The module returns multiple candidates and correspondences for all the languages. If you want to integrate it in your Dutch or Italian pipeline, you will need:
>
> 1. The jar file with the dbpedia-spotlight server. You need the version that Aitor developed in order to correctly use the ”candidates” option. You can copy it from the English VM. The jar file name is `dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar`
> 2. The Dutch/Italian model for the dbpedia-spotlight. You can download them from: http://spotlight.sztaki.hu/downloads/
> 3. The jar file with the NED module: `ixa-pipe-ned-1.0.jar`. You can copy it from the English VM too.
> 4. The file: `wikipedia-db.v1.tar.gz`. You can download it from: http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz. This file contains the required information to do the mappings between the wikipedia-entries. The zip file contains three files: wikipedia-db, wikipedia-db.p and wikipedia-db.t
>
> To start the dbPeadia server: Italian server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar it http://local
```

Dutch server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://local
```

We set 8Gb for the English server, but the Italian and Dutch spotlight will require less memory.

⟨ *install the spotlight server* 16a ⟩ ≡

```
mkdir -p /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/spotlight
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
cp /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/snapshots/spotlight/dbpedia-spotlight-0.7-jar-with-dependencies-
candidates.jar .
```

◇

Fragment defined by 16ab.
Fragment referenced in 10b.

We choose to put the Wikipedia database in the spotlight directory.

⟨ *install the spotlight server* 16b ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
wget http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz
tar -xzf wikipedia-db.v1.tar.gz
rm  wikipedia-db.v1.tar.gz
```

◇

Fragment defined by 16ab.
Fragment referenced in 10b.

⟨ *start the spotlight server* 16c ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-
candidates.jar nl http://localhost:2060/rest  &
```

◇

Fragment referenced in 16d.

⟨ *check/start the spotlight server* 16d ⟩ ≡

```
spottasks=‘netstat -an | grep :2060 | wc -l‘
if
  [ $spottasks -eq 0 ]
then
  ⟨ start the spotlight server 16c ⟩
  sleep 60
fi
```

◇

Fragment referenced in 23f.

3.6 Install modules

3.6.1 Install tokenizer

Module The tokenizer is just a jar that has to be run in Java. Although the jar is directly available from http://ixa2.si.ehu.es/ixa-pipes/download.html, we prefer to compile the package in order to make this thing ready for reproducible set-ups.

To install the tokenizer, we proceed as follows:

1. Clone the source from github into a temporary directory.
2. Compile to produce the jar file with the tokenizer.
3. move the jar file into the jar directory.
4. remove the tempdir with the sourcecode.

⟨ *install the tokenizer* 17a ⟩ ≡
```
tempdir=‘mktemp -d -t tok.XXXXXX‘
cd $tempdir
git clone https://github.com/ixa-ehu/ixa-pipe-tok.git
cd ixa-pipe-tok
mvn clean package
mv target/ixa-pipe-tok-1.6.6.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-
modules-on-Lisa/env/java/jars
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
◇
```
Fragment referenced in 11a.

Script The script runs the tokenizerscript.

"../bin/tok" 17b≡
```
#!/bin/bash
⟨ set up programming environment 4g ⟩
JARFILE=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/java/jars/ixa-pipe-tok-1.6.6.jar
java -jar $JARFILE tok -l nl --inputkaf
◇
```

⟨ *make scripts executable* 17c ⟩ ≡
```
chmod 775  ../bin/tok
◇
```
Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.2 Morphosyntactic parser

Module

⟨ *install the morphosyntactic parser* 17d ⟩ ≡

⟨ *install from github* (17e morphsynparser,17f morphosyntactic_parser_nl,17g https://github.com/cltl/morphosynt
◇
Fragment referenced in 11a.

Script

```
"../bin/mor" 18a≡
        #!/bin/bash
        ⟨ set up programming environment 4g ⟩
        ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
        MODDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
        Lisa/modules/morphosyntactic_parser_nl
        ⟨ set alpinohome 12f ⟩
        cat | python $MODDIR/core/morph_syn_parser.py
        ◇
```

```
⟨ make scripts executable 18b ⟩ ≡
        chmod 775  ../bin/mor
        ◇
```

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.3 Alpino hack

Install a hack that removes output from Alpino that cannot be interpreted by following modules. It is just a small python script. Actually, it may no longer be necessary.

Module

```
⟨ directories to create 18c ⟩ ≡
        ../modules/alpinohack ◇
```

Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.

```
"../modules/alpinohack/clean_hack.py" 18d≡
        #!/usr/bin/python
        import sys

        input = sys.stdin

        output = ''

        for line in input:
            line = line.replace('"--','"#')
            line = line.replace('--"','#"')
            output += line

        print output

        ◇
```

Uses: print 36a.

Script

```
"../bin/alpinohack" 19a≡
        #!/bin/bash
        ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
        HACKDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
        Lisa/modules/alpinohack
        cat | python  $HACKDIR/clean_hack.py

        ◇
```

⟨ *make scripts executable* 19b ⟩ ≡

```
        chmod 775  ../bin/alpinohack
        ◇
```

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.4 Nominal coreference-base

Get this thing from Github (https://github.com/opener-project/coreference-base/) and use the instruction of https://github.com/opener-project/coreference-base/blob/master/core/README.md.

Module

⟨ *install coreference-base* 19c ⟩ ≡

```
        ⟨ install from github (19d coreference-base,19e coreference-base,19f https://github.com/opener-project/corefe
        pip install --upgrade  hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
        pip install --upgrade  networkx
        ◇
```

Fragment referenced in 11a.

Script

```
"../bin/coreference-base" 19g≡
        #!/bin/bash
        ⟨ set up programming environment 4g ⟩
        cd $PIPEMODD/coreference-base/core
        cat | python -m corefgraph.process.file --language nl --singleton --sieves NO
        ◇
```

⟨ *make scripts executable* 19h ⟩ ≡

```
        chmod 775  ../bin/coreference-base
        ◇
```

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.5 Named entity recognition (NERC)

Module The Nerc program can be installed from Github (https://github.com/ixa-ehu/ixa-pipe-nerc). However, the model that is needed is not publicly available. Therefore, the Nerc module of the standard English pipeline, that is not yet public available, has been put in the snapshot-tarball.

⟨ *install the NERC module* 20a ⟩ ≡
⟨ *compile the nerc jar* 20b ⟩
⟨ *get the nerc models* 20c ⟩

```
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/EHU-
nerc  /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/
```

◇

Fragment referenced in 11a.

⟨ *compile the nerc jar* 20b ⟩ ≡

```
TEMPDIR=‘mktemp -d -t nerc.XXXXXX‘
cd $TEMPDIR
git clone https://github.com/ixa-ehu/ixa-pipe-nerc
cd ixa-pipe-nerc/
mvn clean package
mv target/ixa-pipe-nerc-1.3.3.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-
modules-on-Lisa/env/java/jars/
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/nuweb
rm -rf $TEMPDIR
```

◇

Fragment referenced in 20a.
Uses: nuweb 38b.

⟨ *get the nerc models* 20c ⟩ ≡

```
mkdir -p ../modules/EHU-nerc
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/EHU-
nerc/nerc-resources ../modules/EHU-nerc
```

◇

Fragment referenced in 20a.

Script

"../bin/nerc" 20d≡

```
#!/bin/bash
```

⟨ *set up programming environment* 4g ⟩

```
MODDIR=$PIPEMODD/EHU-nerc
JAR=$JARDIR/ixa-pipe-nerc-1.3.3.jar
MODEL=nl-local-conll02-testa.bin
cat | java -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL
```

◇

⟨ *make scripts executable* 20e ⟩ ≡

```
chmod 775  ../bin/nerc
```

◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.6 Wordsense-disambiguation

Install WSD from its Github source (https://github.com/cltl/svm_wsd.git). According to the `readme` of that module, the next thing to do is, to execute install-script `install.sh` or `install_naf.sh`. The latter script installs a "Support-Vector-Machine" (SVM) module, "Dutch-SemCor" (DSC) models and KafNafParserPy.

Module

⟨ *install the WSD module* 21a ⟩ ≡
```
⟨ install from github (21b wsd,21c svm_wsd,21d https://github.com/cltl/svm_wsd.git ) 9d ⟩
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/svm_wsd
⟨ install svm lib 21e ⟩
⟨ download svm models 21f ⟩


◇
```
Fragment referenced in 11a.

This part has been copied from `install_naf.sh` in the WSD module.

⟨ *install svm lib* 21e ⟩ ≡
```
mkdir lib
cd lib
wget --no-check-
certificate  https://github.com/cjlin1/libsvm/archive/master.zip 2>/dev/null
zip_name=‘ls -1 | head -1‘
unzip $zip_name > /dev/null
rm $zip_name
folder_name=‘ls -1 | head -1‘
mv $folder_name libsvm
cd libsvm/python
make > /dev/null 2> /dev/null
echo LIBSVM installed correctly lib/libsvm
◇
```
Fragment referenced in 21a.

This part has also been copied from `install_naf.sh` in the WSD module.

⟨ *download svm models* 21f ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/svm_wsd
echo 'Downloading models...(could take a while)'
wget --user=cltl --
password='.cltl.' kyoto.let.vu.nl/~izquierdo/models_wsd_svm_dsc.tgz 2> /dev/null
echo 'Unzipping models...'
tar xzf models_wsd_svm_dsc.tgz
rm models_wsd_svm_dsc.tgz
echo 'Models installed in folder models'


◇
```
Fragment referenced in 21a.

Script

"../bin/wsd" 22a≡
```
#!/bin/bash
# WSD -- wrapper for word-sense disambiguation
# 8 Jan 2014 Ruben Izquierdo
# 16 sep 2014 Paul Huygen
⟨ set up programming environment 4g ⟩
WSDDIR=$PIPEMODD/svm_wsd
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $WSDDIR/$WSDSCRIPT --naf
◇
```

⟨ make scripts executable 22b ⟩ ≡
```
chmod 775  ../bin/wsd
◇
```
Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

2

3.6.7 Lexical-unit converter

Module There is not an official repository for this module yet, so copy the module from the tarball.

⟨ install the lu2synset converter 22c ⟩ ≡
```
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/snapshots/lexicalunitconvertor /home/paul/projecten/cltl/pipelines/dutch-nlp-
modules-on-Lisa/modules/
◇
```
Fragment never referenced.

Script

"../bin/lu2synset" 22d≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
JAVALIBDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/lexicalunitconvertor/lib
RESOURCESDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/lexicalunitconvertor/resources
JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
java -Xmx812m -
cp  $JAVALIBDIR/$JARFILE vu.wntools.util.NafLexicalUnitToSynsetReferences \
   --wn-lmf "$RESOURCESDIR/cornetto2.1.lmf.xml" --format naf
◇
```

3.6.8 NED

The NED module is rather picky about the structure of the NAF file. In any case, it does not accept a file that has been produced by the ontotagger. Hence, in a pipeline NER shuld be executed before the ontotagger.

The NED module wants to consult the dbpedia spotlight server, so that one has to be installed somewhere. For this moment, let us suppose that it has been installed on localhost.

Module

⟨ *install the* NED *module* 23a ⟩ ≡
⟨ *put spotlight jar in the Maven repository* 23e ⟩

⟨ *install from github* (23b `ned`,23c `ixa-pipe-ned`,23d `https://github.com/ixa-ehu/ixa-pipe-ned.git`) 9d ⟩
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/ixa-pipe-
ned
mvn -Dmaven.compiler.target=1.7 -Dmaven.compiler.source=1.7 clean package
mv target/ixa-pipe-ned-1.1.1.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-
modules-on-Lisa/env/java/jars/
```
◇

Fragment referenced in 11a.

NED needs to have dbpedia-spotlight-0.7.jar in the local Maven repository. That is a different jar than the jar that we use to start Spotlight.

⟨ *put spotlight jar in the Maven repository* 23e ⟩ ≡
```
tempdir=‘mktemp -d -t simplespot.XXXXXX‘
cd $tempdir
wget http://spotlight.sztaki.hu/downloads/dbpedia-spotlight-0.7.jar


cd $PROJROOT
rm -rf $tempdir
```
◇

Fragment referenced in 23a.

Script

"../bin/ned" 23f≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
JARDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java/jars
```
⟨ *check/start the spotlight server* 16d ⟩
```
cat | java -jar $JARDIR/ixa-pipe-ned-1.1.1.jar  -p 2060 -e candidates -
i /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/env/spotlight/wikipedia-db -n nlEn
```
◇

⟨ *make scripts executable* 23g ⟩ ≡
```
chmod 775  ../bin/ned
```
◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.9 Ontotagger

We do not yet have a source-repository of the Ontotagger module. Therefore, install from a snapshot (`vua-ontotagger-v1.0.tar.gz`).

Module

⟨ *install the onto module* 24a ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
tar -xzf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/snapshots/vua-ontotagger-v1.0.tar.gz
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/vua-
ontotagger-v1.0 /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/
chmod -R o+r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
◇
```

Fragment referenced in 11a.

Script

"../bin/onto" 24b≡

```
#!/bin/bash
⟨ set up programming environment 4g ⟩
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
ONTODIR=$PIPEMODD/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=‘mktemp -t stap6.XXXXXX‘
cat >$TMPFIL

CLASSPATH=$JARDIR/ontotagger-1.0-jar-with-dependencies.jar
JAVASCRIPT=eu.kyotoproject.main.KafPredicateMatrixTagger

MAPPINGS="fn;mcr;ili;eso"
JAVA_ARGS="--mappings $MAPPINGS"
JAVA_ARGS="$JAVA_ARGS  --key odwn-eq"
JAVA_ARGS="$JAVA_ARGS  --version 1.1"
JAVA_ARGS="$JAVA_ARGS  --predicate-matrix $PREDICATEMATRIX"
JAVA_ARGS="$JAVA_ARGS  --grammatical-words $GRAMMATICALWORDS"
JAVA_ARGS="$JAVA_ARGS  --naf-file $TMPFIL"
java -Xmx1812m -cp $CLASSPATH $JAVASCRIPT $JAVA_ARGS




rm -rf $TMPFIL

◇
```

⟨ *make scripts executable* 25a ⟩ ≡
```
chmod 775  ../bin/onto
```
◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.10 Framenet SRL

The framenet SRL is part of the package that contains the ontotagger. We only need a different script.

Script The script contains a hack, because the framesrl script produces spiruous lines containint "frameMap.size()=...". A GAWK script removes these lines.

"../bin/framesrl" 25b≡
```
#!/bin/bash
⟨ set up programming environment 4g ⟩
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
ONTODIR=$PIPEMODD/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=‘mktemp -t framesrl.XXXXXX‘
cat >$TMPFIL

CLASSPATH=$JARDIR/ontotagger-1.0-jar-with-dependencies.jar
JAVASCRIPT=eu.kyotoproject.main.SrlFrameNetTagger

JAVA_ARGS="--naf-file $TMPFIL"
JAVA_ARGS="$JAVA_ARGS  --format naf"
JAVA_ARGS="$JAVA_ARGS  --frame-ns fn:"
JAVA_ARGS="$JAVA_ARGS   --role-ns fn-role:;pb-role:;fn-pb-role:;eso-role:"
JAVA_ARGS="$JAVA_ARGS   --ili-ns mcr:ili"
JAVA_ARGS="$JAVA_ARGS   --sense-conf 0.25"
JAVA_ARGS="$JAVA_ARGS   --frame-conf 70"

java -Xmx1812m -
cp $CLASSPATH $JAVASCRIPT $JAVA_ARGS  | gawk '/^frameMap.size()/ {next}; {print}'



rm -rf $TMPFIL

◇
```
Uses: print 36a.

⟨ *make scripts executable* 25c ⟩ ≡
```
chmod 775  ../bin/framesrl
```
◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.11 Heideltime

Module

⟨ *install the heideltime module* 26a ⟩ ≡

⟨ *install from github* (26b heideltime,26c NAF-HeidelTime,26d git@github.com:cltl/NAF-HeidelTime.git) 9d ⟩
⟨ *adapt heideltime's config.props* 26e ⟩

◇

Fragment referenced in 11a.

⟨ *adapt heideltime's config.props* 26e ⟩ ≡

```
CONFIL=NAF-HeidelTime/config.props
tempfil=‘mktemp -t heideltmp.XXXXXX‘
mv $CONFIL $tempfil
MODDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
TREETAGDIR=treetagger
AWKCOMMAND=’/ˆtreeTaggerHome/ {$0="treeTagger-
Home = /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/treetagger"}; {print}’
gawk "$AWKCOMMAND" $tempfil >$CONFIL
```

◇

Fragment referenced in 26a.
Uses: print 36a.

Script

"../bin/heideltime" 26f≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
HEIDELDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
Lisa/modules/NAF-HeidelTime
TEMPDIR=‘mktemp -t -d heideltmp.XXXXXX‘
cd $HEIDELDIR
```

⟨ *activate the python environment* 7c, . . . ⟩

```
iconv -t utf-
8//IGNORE | python $HEIDELDIR/HeidelTime_NafKaf.py $HEIDELDIR/heideltime-
standalone/ $TEMPDIR
```

◇

⟨ *make scripts executable* 26g ⟩ ≡

```
chmod 775  ../bin/heideltime
```

◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.12 Semantic Role labelling

Module

⟨ *install the srl module* 27a ⟩ ≡

```
⟨ install from github (27b srl,27c vua-srl-nl,27d https://github.com/newsreader/vua-srl-nl.git ) 9d ⟩
◇
```

Fragment referenced in 11a.

Script First:

1. set the correct environment. The module needs python and timble.
2. create a tempdir and in that dir a file to store the input and a (SCV) file with the feature-vector.

"../bin/srl" 27e≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
SRLDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/vua-
srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $SRLDIR
⟨ set local bin directory 4e ⟩
⟨ activate the python environment 7c, ... ⟩
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
◇
```

File defined by 27efgh, 28a.

Create a feature-vector.

"../bin/srl" 27f≡

```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
◇
```

File defined by 27efgh, 28a.

Run the trained model on the feature-vector.

"../bin/srl" 27g≡

```
timbl -mO:I1,2,3,4 -i e-mags_mags_press_newspapers.wgt -t $FEATUREVECTOR -
o $TIMBLOUTPUTFILE >/dev/null 2>/dev/null
◇
```

File defined by 27efgh, 28a.

Insert the SRL values into the NAF file.

"../bin/srl" 27h≡

```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
◇
```

File defined by 27efgh, 28a.

Clean up.

```
"../bin/srl" 28a≡
        rm -rf $TEMPDIR
        ◇
```

File defined by 27efgh, 28a.

⟨ *make scripts executable* 28b ⟩ ≡

```
        chmod 775  ../bin/srl
        ◇
```

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

3.6.13 Event coreference

Module Install the module from the snapshot.

⟨ *install the event-coreference module* 28c ⟩ ≡

```
        cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
        tar -xzf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
        Lisa/snapshots/vua-eventcoreference_v2.tar.gz
        cd vua-eventcoreference_v2
        cp lib/EventCoreference-1.0-SNAPSHOT-jar-with-
        dependencies.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
        Lisa/env/java/jars
        ◇
```

Fragment referenced in 11a.

Script

```
"../bin/evcoref" 28d≡
        #!/bin/bash
        ⟨ set up programming environment 4g ⟩
        MODROOT=$PIPEMODD/vua-eventcoreference_v2
        RESOURCESDIR=$MODROOT/resources
        JARFILE=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-
        Lisa/env/java/jars/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar

        JAVAMODULE=eu.newsreader.eventcoreference.naf.EventCorefWordnetSim
        JAVAOPTIONS="--method leacock-chodorow"
        JAVAOPTIONS="$JAVAOPTIONS  --wn-lmf $RESOURCESDIR/cornetto2.1.lmf.xml"
        JAVAOPTIONS="$JAVAOPTIONS  --sim 2.0"
        JAVAOPTIONS="$JAVAOPTIONS  --
        relations XPOS_NEAR_SYNONYM#HAS_HYPERONYM#HAS_XPOS_HYPERONYM"


        java -Xmx812m -cp $JARFILE $JAVAMODULE  $JAVAOPTIONS

        ◇
```

⟨ *make scripts executable* 29a ⟩ ≡
```
chmod 775  ../bin/evcoref
```
◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

4 Utilities

4.1 Test script

The following script pushes a single sentence through the modules of the pipeline.

"../bin/test" 29b≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
TESTDIR=$ROOT/test
BIND=$ROOT/bin
mkdir -p $TESTDIR
cd $TESTDIR
cat $ROOT/nuweb/testin.naf | $BIND/tok > $TESTDIR/test.tok.naf
cat test.tok.naf | $BIND/mor > $TESTDIR/test.mor.naf
cat test.mor.naf | $BIND/nerc > $TESTDIR/test.nerc.naf
cat $TESTDIR/test.nerc.naf | $BIND/wsd > $TESTDIR/test.wsd.naf
cat $TESTDIR/test.wsd.naf | $BIND/ned  > $TESTDIR/test.ned.naf
cat $TESTDIR/test.ned.naf | $BIND/onto > $TESTDIR/test.onto.naf
cat $TESTDIR/test.onto.naf | $BIND/heideltime > $TESTDIR/test.times.naf
cat $TESTDIR/test.times.naf | $BIND/srl  > $TESTDIR/test.srl.naf
cat $TESTDIR/test.srl.naf | $BIND/evcoref  > $TESTDIR/test.ecrf.naf
cat $TESTDIR/test.ecrf.naf | $BIND/framesrl  > $TESTDIR/test.fsrl.naf

```
◇

Uses: nuweb 38b.

⟨ *make scripts executable* 29c ⟩ ≡
```
chmod 775  ../bin/test
```
◇

Fragment defined by 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac.
Fragment referenced in 42c.

4.2 Logging

Write log messages to standard out if variable LOGLEVEL is equal to 1.

⟨ *variables of install-modules* 29d ⟩ ≡
```
LOGLEVEL=1
```
◇

Fragment referenced in 10b.

⟨ *logmess* 30a ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
```
◇

Fragment referenced in 8a, 9bd, 12a, 15a, 30b.

4.3 Misc

Install a module from a tarball: The macro expects the following three variables to be present:

URL: The URL tfrom where the taball can be downloaded.
TARB: The name of the tarball.
DIR; Name of the directory for the module.

Arg 1: URL; Arg 2: tarball; Arg 3: directory.

⟨ *install from tarball* 30b ⟩ ≡
```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (30c $DIR ) 8g ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (30d Installed $DIR ) 30a ⟩
  ⟨ remove old module (30e $DIR ) 9a ⟩
else
  ⟨ re-instate old module (30f $DIR ) 9b ⟩
fi
```
◇

Fragment never referenced.

A How to read and translate this document

This document is an example of *literate programming* [1]. It contains the code of all sorts of scripts and programs, combined with explaining texts. In this document the literate programming tool `nuweb` is used, that is currently available from Sourceforge (URL:`nuweb.sourceforge.net`). The advantages of Nuweb are, that it can be used for every programming language and scripting language, that it can contain multiple program sources and that it is very simple.

A.1 Read this document

The document contains *code scraps* that are collected into output files. An output file (e.g. `output.fil`) shows up in the text as follows:

"output.fil" 4a ≡
```
# output.fil
< a macro 4b >
< another macro 4c >
◇
```

The above construction contains text for the file. It is labelled with a code (in this case 4a) The constructions between the < and > brackets are macro's, placeholders for texts that can be found in other places of the document. The test for a macro is found in constructions that look like:

< a macro 4b > ≡
```
This is a scrap of code inside the macro.
It is concatenated with other scraps inside the
macro. The concatenated scraps replace
the invocation of the macro.
```

```
Macro defined by 4b, 87e
Macro referenced in 4a
```

Macro's can be defined on different places. They can contain other macro's.

< a scrap 87e > ≡
```
This is another scrap in the macro. It is
concatenated to the text of scrap 4b.
This scrap contains another macro:
< another macro 45b >
```

```
Macro defined by 4b, 87e
Macro referenced in 4a
```

A.2 Process the document

The raw document is named `a_dutch-nlp-modules-on-Lisa.w`. Figure 1 shows pathways to trans-

Figure 1: *Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.*

late it into printable/viewable documents and to extract the program sources. Table 3 lists the

Tool	Source	Description
gawk	www.gnu.org/software/gawk/	text-processing scripting language
M4	www.gnu.org/software/m4/	Gnu macro processor
nuweb	nuweb.sourceforge.net	Literate programming tool
tex	www.ctan.org	Typesetting system
tex4ht	www.ctan.org	Convert TEX documents into `xml`/`html`

Table 3: *Tools to translate this document into readable code and to extract the program sources*

tools that are needed for a translation. Most of the tools (except Nuweb) are available on a well-equipped Linux system.

A.3 Translate and run

This chapter assembles the Makefile for this project.

```
"Makefile" 32a≡
    ⟨ default target 32b ⟩

    ⟨ parameters in Makefile 32d, ... ⟩

    ⟨ impliciete make regels 35a, ... ⟩
    ⟨ expliciete make regels 33a, ... ⟩
    ⟨ make targets 36a, ... ⟩
    ◇
```

The default target of make is `all`.

```
⟨ default target 32b ⟩ ≡
    all : ⟨ all targets 32c ⟩
    .PHONY : all

    ◇
```

Fragment referenced in 32a.
Defines: `all` Never used, `PHONY` 35b.

One of the targets is certainly the PDF version of this document.

```
⟨ all targets 32c ⟩ ≡
    dutch-nlp-modules-on-Lisa.pdf◇
```

Fragment referenced in 32b.
Uses: `pdf` 36a.

We use many suffixes that were not known by the C-programmers who constructed the `make` utility. Add these suffixes to the list.

```
⟨ parameters in Makefile 32d ⟩ ≡
    .SUFFIXES: .pdf .w .tex .html .aux .log .php

    ◇
```

Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 36a.

A.4 Get Nuweb

An annoying problem is, that this program uses nuweb, a utility that is seldom installed on a computer. Therefore, we are going to install that first if it is not present. Unfortunately, nuweb is hosted on sourceforge and it is difficult to achieve automatic downloading from that repository. Therefore I copied one of the versions on a location from where it can be downloaded with a script.

```
⟨ parameters in Makefile 32e ⟩ ≡
    NUWEB=../bin/nuweb
    ◇
```

Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Defines: `NUWEB` 33a, 37abc, 38b, 40c, 41c, 42c.
Uses: `nuweb` 38b.

⟨ *expliciete make regels* 33a ⟩ ≡

```
$(NUWEB): ../nuweb-1.58
        cd ../nuweb-1.58 && make nuweb
        cp ../nuweb-1.58/nuweb $(NUWEB)
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: NUWEB 32e, nuweb 38b.

⟨ *expliciete make regels* 33b ⟩ ≡

```
../nuweb-1.58:
        cd .. && wget http://kyoto.let.vu.nl/~huygen/nuweb-1.58.tgz
        cd .. &&  tar -xzf nuweb-1.58.tgz
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: nuweb 38b.

A.5 Pre-processing

To make usable things from the raw input `a_dutch-nlp-modules-on-Lisa.w`, do the following:

1. Process `$` characters.
2. Run the m4 pre-processor.
3. Run nuweb.

This results in a LaTeX file, that can be converted into a PDF or a HTML document, and in the program sources and scripts.

A.5.1 Process 'dollar' characters

Many "intelligent" TeX editors (e.g. the auctex utility of Emacs) handle `$` characters as special, to switch into mathematics mode. This is irritating in program texts, that often contain `$` characters as well. Therefore, we make a stub, that translates the two-character sequence `\$` into the single `$` character.

⟨ *expliciete make regels* 33c ⟩ ≡

```
m4_dutch-nlp-modules-on-Lisa.w : a_dutch-nlp-modules-on-Lisa.w
        gawk '{if(match($$0, "@%")) {printf("%s", substr($$0,1,RSTART-
1))} else print}' a_dutch-nlp-modules-on-Lisa.w \
          | gawk '{gsub(/[\\][\$$]/, "$$");print}'  > m4_dutch-nlp-modules-on-
Lisa.w
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: print 36a.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 34a ⟩ ≡

```
dutch-nlp-modules-on-Lisa.w : m4_dutch-nlp-modules-on-Lisa.w inst.m4
        m4 -P m4_dutch-nlp-modules-on-Lisa.w > dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.

A.6 Typeset this document

Enable the following:

1. Create a PDF document.
2. Print the typeset document.
3. View the typeset document with a viewer.
4. Create a HTMLdocument.

In the three items, a typeset PDF document is required or it is the requirement itself.

A.6.1 Figures

This document contains figures that have been made by `xfig`. Post-process the figures to enable inclusion in this document.

The list of figures to be included:

⟨ *parameters in Makefile* 34b ⟩ ≡

```
FIGFILES=fileschema
```

◇

Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Defines: `FIGFILES` 34c, 39b.

We use the package `figlatex` to include the pictures. This package expects two files with extensions `.pdftex` and `.pdftex_t` for `pdflatex` and two files with extensions `.pstex` and `.pstex_t` for the `latex/dvips` combination. Probably tex4ht uses the latter two formats too.

Make lists of the graphical files that have to be present for latex/pdflatex:

⟨ *parameters in Makefile* 34c ⟩ ≡

```
FIGFILENAMES=$(foreach fil,$(FIGFILES), $(fil).fig)
PDFT_NAMES=$(foreach fil,$(FIGFILES), $(fil).pdftex_t)
PDF_FIG_NAMES=$(foreach fil,$(FIGFILES), $(fil).pdftex)
PST_NAMES=$(foreach fil,$(FIGFILES), $(fil).pstex_t)
PS_FIG_NAMES=$(foreach fil,$(FIGFILES), $(fil).pstex)
```

◇

Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 36b, `PDF_FIG_NAMES` 36b, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 34b.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 35a ⟩ ≡

```
%.eps: %.fig
        fig2dev -L eps $< > $@

%.pstex: %.fig
        fig2dev -L pstex $< > $@

.PRECIOUS : %.pstex
%.pstex_t: %.fig %.pstex
        fig2dev -L pstex_t -p $*.pstex $< > $@

%.pdftex: %.fig
        fig2dev -L pdftex $< > $@

.PRECIOUS : %.pdftex
%.pdftex_t: %.fig %.pstex
        fig2dev -L pdftex_t -p $*.pdftex $< > $@
```

◇

Fragment defined by 35a, 36b, 39c.
Fragment referenced in 32a.
Defines: `fig2dev` Never used.

A.6.2 Bibliography

To keep this document portable, create a portable bibliography file. It works as follows: This document refers in the |bibliography| statement to the local `bib`-file `dutch-nlp-modules-on-Lisa.bib`. To create this file, copy the auxiliary file to another file `auxfil.aux`, but replace the argument of the command `\bibdata{dutch-nlp-modules-on-Lisa}` to the names of the bibliography files that contain the actual references (they should exist on the computer on which you try this). This procedure should only be performed on the computer of the author. Therefore, it is dependent of a binary file on his computer.

⟨ *expliciete make regels* 35b ⟩ ≡

```
bibfile : dutch-nlp-modules-on-Lisa.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib dutch-nlp-modules-on-Lisa litprog

.PHONY : bibfile
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: `PHONY` 32b.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 36a ⟩ ≡
```
pdf : dutch-nlp-modules-on-Lisa.pdf

print : dutch-nlp-modules-on-Lisa.pdf
        lpr dutch-nlp-modules-on-Lisa.pdf

view : dutch-nlp-modules-on-Lisa.pdf
        evince dutch-nlp-modules-on-Lisa.pdf

◇
```
Fragment defined by 36a, 39a, 42bc.
Fragment referenced in 32a.
Defines: `pdf` 32cd, 36b, `print` 18d, 25b, 26e, 33c, `view` Never used.

Create the PDF document. This may involve multiple runs of nuweb, the LaTeX processor and the bibTeX processor, and depends on the state of the `aux` file that the LaTeX processor creates as a by-product. Therefore, this is performed in a separate script, `w2pdf`.

The w2pdf script The three processors nuweb, LaTeX and bibTeX are intertwined. LaTeX and bibTeX create parameters or change the value of parameters, and write them in an auxiliary file. The other processors may need those values to produce the correct output. The LaTeX processor may even need the parameters in a second run. Therefore, consider the creation of the (PDF) document finished when none of the processors causes the auxiliary file to change. This is performed by a shell script `w2pdf`.

Note, that in the following `make` construct, the implicit rule `.w.pdf` is not used. It turned out, that make did not calculate the dependencies correctly when I did use this rule.

⟨ *impliciete make regels* 36b ⟩ ≡
```
%.pdf : %.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

◇
```
Fragment defined by 35a, 36b, 39c.
Fragment referenced in 32a.
Uses: `pdf` 36a, `PDFT_NAMES` 34c, `PDF_FIG_NAMES` 34c.

The following is an ugly fix of an unsolved problem. Currently I develop this thing, while it resides on a remote computer that is connected via the `sshfs` filesystem. On my home computer I cannot run executables on this system, but on my work-computer I can. Therefore, place the following script on a local directory.

⟨ *directories to create* 36c ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 3ab, 4abcd, 5af, 6b, 7d, 18c, 36c.
Fragment referenced in 42b.
Uses: `nuweb` 38b.

⟨ *parameters in Makefile* 36d ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Uses: `nuweb` 38b.

⟨ *expliciete make regels* 37a ⟩ ≡

```
$(W2PDF) : dutch-nlp-modules-on-Lisa.w $(NUWEB)
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: NUWEB 32e.

"../nuweb/bin/w2pdf" 37b≡

```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20150130 at 1236h: Generated by nuweb from a_dutch-nlp-modules-on-Lisa.w
NUWEB=/usr/local/bin/nuweb

LATEXCOMPILER=pdflatex
⟨ filenames in nuweb compile script 37d ⟩
⟨ compile nuweb 37c ⟩

◇
```

Uses: NUWEB 32e, nuweb 38b.

The script retains a copy of the latest version of the auxiliary file. Then it runs the four processors nuweb, LaTeX, MakeIndex and bibTeX, until they do not change the auxiliary file or the index.

⟨ *compile nuweb* 37c ⟩ ≡

```
NUWEB=/usr/local/bin/nuweb

⟨ run the processors until the aux file remains unchanged 38c ⟩
⟨ remove the copy of the aux file 38a ⟩
◇
```

Fragment referenced in 37b.
Uses: NUWEB 32e, nuweb 38b.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. `.w`) from the filename and create the names of the LaTeX file (ends with `.tex`), the auxiliary file (ends with `.aux`) and the copy of the auxiliary file (add `old.` as a prefix to the auxiliary filename).

⟨ *filenames in nuweb compile script* 37d ⟩ ≡

```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```

Fragment referenced in 37b.
Defines: auxfil 38c, 41ab, indexfil 38c, 41a, nufil 38b, 41ac, oldaux 38ac, 41ab, oldindexfil 38c, 41a, texfil 38b, 41ac, trunk 38b, 41acd.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 38a ⟩ ≡
```
rm $oldaux
```
◇

Fragment referenced in 37c, 40d.
Uses: `oldaux` 37d, 41a.

Run the three processors. Do not use the option `-o` (to suppres generation of program sources) for nuweb, because `w2pdf` must be kept up to date as well.

⟨ *run the three processors* 38b ⟩ ≡
```
$NUWEB $nufil
$LATEXCOMPILER $texfil
makeindex $trunk
bibtex $trunk
```
◇

Fragment referenced in 38c.
Defines: `bibtex` 41cd, `makeindex` 41cd, `nuweb` 20b, 29b, 32e, 33ab, 36cd, 37bc, 39a, 40bc.
Uses: `nufil` 37d, 41a, `NUWEB` 32e, `texfil` 37d, 41a, `trunk` 37d, 41a.

Repeat to copy the auxiliary file and the index file and run the processors until the auxiliary file and the index file are equal to their copies. However, since I have not yet been able to test the `aux` file and the `idx` in the same test statement, currently only the `aux` file is tested.

It turns out, that sometimes a strange loop occurs in which the `aux` file will keep to change. Therefore, with a counter we prevent the loop to occur more than 10 times.

⟨ *run the processors until the aux file remains unchanged* 38c ⟩ ≡
```
LOOPCOUNTER=0
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  if [ -e $indexfil ]
  then
   cp $indexfil $oldindexfil
  fi
  ⟨ run the three processors 38b ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
```
◇

Fragment referenced in 37c.
Uses: `auxfil` 37d, 41a, `indexfil` 37d, `oldaux` 37d, 41a, `oldindexfil` 37d.

A.6.4 Create HTML files

HTML is easier to read on-line than a PDF document that was made for printing. We use `tex4ht` to generate HTML code. An advantage of this system is, that we can include figures in the same way as we do for `pdflatex`.

Nuweb creates a LaTeX file that is suitable for `latex2html` if the source file has `.hw` as suffix instead of `.w`. However, this feature is not compatible with tex4ht.

Make html file:

⟨ *make targets* 39a ⟩ ≡
```
html : ../nuweb/html/dutch-nlp-modules-on-Lisa.html

```
◇
Fragment defined by 36a, 39a, 42bc.
Fragment referenced in 32a.
Uses: `nuweb` 38b.

The HTML file depends on its source file and the graphics files.

Make lists of the graphics files and copy them.

⟨ *parameters in Makefile* 39b ⟩ ≡
```
HTML_PS_FIG_NAMES=$(foreach fil,$(FIGFILES), m4_htmldocdir/$(fil).pstex)
HTML_PST_NAMES=$(foreach fil,$(FIGFILES), m4_htmldocdir/$(fil).pstex_t)
```
◇
Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Uses: `FIGFILES` 34b.

⟨ *impliciete make regels* 39c ⟩ ≡
```
m4_htmldocdir/%.pstex : %.pstex
        cp  $< $@

m4_htmldocdir/%.pstex_t : %.pstex_t
        cp  $< $@

```
◇
Fragment defined by 35a, 36b, 39c.
Fragment referenced in 32a.

Copy the nuweb file into the html directory.

⟨ *expliciete make regels* 39d ⟩ ≡
```
m4_htmlsource : dutch-nlp-modules-on-Lisa.w
        cp  dutch-nlp-modules-on-Lisa.w m4_htmlsource

```
◇
Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.

We also need a file with the same name as the documentstyle and suffix `.4ht`. Just copy the file `report.4ht` from the tex4ht distribution. Currently this seems to work.

⟨ *expliciete make regels* 39e ⟩ ≡
```
m4_4htfildest : m4_4htfilsource
        cp m4_4htfilsource m4_4htfildest

```
◇
Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.

Copy the bibliography.

⟨ *expliciete make regels* 40a ⟩ ≡
```
m4_htmlbibfil : m4_nuwebdir/dutch-nlp-modules-on-Lisa.bib
        cp m4_nuwebdir/dutch-nlp-modules-on-Lisa.bib m4_htmlbibfil
```
◇
Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.

Make a dvi file with `w2html` and then run `htlatex`.

⟨ *expliciete make regels* 40b ⟩ ≡
```
../nuweb/html/dutch-nlp-modules-on-
Lisa.html : m4_htmlsource m4_4htfildest $(HTML_PS_FIG_NAMES) $(HTML_PST_NAMES) m4_htmlbibfil
        cp w2html ../bin
        cd ../bin && chmod 775 w2html
        cd m4_htmldocdir && ../bin/w2html dutch-nlp-modules-on-Lisa.w
```
◇
Fragment defined by 33abc, 34a, 35b, 37a, 39de, 40ab.
Fragment referenced in 32a.
Uses: `nuweb` 38b.

Create a script that performs the translation.

`"w2html"` 40c≡
```
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
#  [filename]: Name of the nuweb source file.
‘#’ m4_header
echo "translate " $1 >w2html.log
NUWEB=/usr/local/bin/nuweb

⟨ filenames in w2html 41a ⟩

⟨ perform the task of w2html 40d ⟩
```
◇
Uses: `NUWEB` 32e, `nuweb` 38b.

The script is very much like the `w2pdf` script, but at this moment I have still difficulties to compile the source smoothly into HTML and that is why I make a separate file and do not recycle parts from the other file. However, the file works similar.

⟨ *perform the task of w2html* 40d ⟩ ≡
```
⟨ run the html processors until the aux file remains unchanged 41b ⟩
⟨ remove the copy of the aux file 38a ⟩
```
◇
Fragment referenced in 40c.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. `.w`) from the filename and create the names of the LaTeX file (ends with `.tex`), the auxiliary file (ends with `.aux`) and the copy of the auxiliary file (add `old.` as a prefix to the auxiliary filename).

⟨ *filenames in w2html* 41a ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 40c.
Defines: `auxfil` 37d, 38c, 41b, `nufil` 37d, 38b, 41c, `oldaux` 37d, 38ac, 41b, `texfil` 37d, 38b, 41c, `trunk` 37d, 38b, 41cd.
Uses: `indexfil` 37d, `oldindexfil` 37d.

⟨ *run the html processors until the aux file remains unchanged* 41b ⟩ ≡
```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 41c ⟩
done
⟨ run tex4ht 41d ⟩

◇
```
Fragment referenced in 40d.
Uses: `auxfil` 37d, 41a, `oldaux` 37d, 41a.

To work for HTML, nuweb *must* be run with the `-n` option, because there are no page numbers.

⟨ *run the html processors* 41c ⟩ ≡
```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 41b.
Uses: `bibtex` 38b, `makeindex` 38b, `nufil` 37d, 41a, `NUWEB` 32e, `texfil` 37d, 41a, `trunk` 37d, 41a.

When the compilation has been satisfied, run makeindex in a special way, run bibtex again (I don't know why this is necessary) and then run htlatex another time.

⟨ *run tex4ht* 41d ⟩ ≡
```
tex '\def\filename{{dutch-nlp-modules-on-Lisa}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 41b.
Uses: `bibtex` 38b, `makeindex` 38b, `trunk` 37d, 41a.

create the program sources Run nuweb, but suppress the creation of the LaTeX documentation. Nuweb creates only sources that do not yet exist or that have been modified. Therefore make does not have to check this. However, "make" has to create the directories for the sources if they do not yet exist. So, let's create the directories first.

⟨ *parameters in Makefile* 42a ⟩ ≡
```
MKDIR = mkdir -p
```
◇
Fragment defined by 32de, 34bc, 36d, 39b, 42a.
Fragment referenced in 32a.
Defines: MKDIR 42b.

⟨ *make targets* 42b ⟩ ≡
```
DIRS = ⟨ directories to create 3a, ... ⟩

$(DIRS) :
        $(MKDIR) $@
```
◇
Fragment defined by 36a, 39a, 42bc.
Fragment referenced in 32a.
Defines: DIRS 42c.
Uses: MKDIR 42a.

⟨ *make targets* 42c ⟩ ≡
```
sources : dutch-nlp-modules-on-Lisa.w $(DIRS) $(NUWEB)
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
        ⟨ make scripts executable 11c, ... ⟩
```
◇
Fragment defined by 36a, 39a, 42bc.
Fragment referenced in 32a.
Uses: DIRS 42b, NUWEB 32e.

B References

B.1 Literature

References

[1] Donald E. Knuth. Literate programming. Technical report STAN-CS-83-981, Stanford University, Department of Computer Science, 1983.

B.2 URL's

Nuweb: nuweb.sourceforge.net
Apache Velocity: m4_velocityURL
Velocitytools: m4_velocitytoolsURL
Parameterparser tool: m4_parameterparserdocURL
Cookietool: m4_cookietooldocURL
VelocityView: m4_velocityviewURL

VelocityLayoutServlet: m4_velocitylayoutservletURL
Jetty: m4_jettycodehausURL
UserBase javadoc: m4_userbasejavadocURL
VU corpus Management development site: http://code.google.com/p/vucom

C Indexes

C.1 Filenames

C.2 Macro's

⟨ install the heideltime module 26a ⟩ Referenced in 11a.
⟨ install the lu2synset converter 22c ⟩ Not referenced.
⟨ install the morphosyntactic parser 17d ⟩ Referenced in 11a.
⟨ install the NERC module 20a ⟩ Referenced in 11a.
⟨ install the onto module 24a ⟩ Referenced in 11a.
⟨ install the spotlight server 16ab ⟩ Referenced in 10b.
⟨ install the srl module 27a ⟩ Referenced in 11a.
⟨ install the ticcutils utility 14c ⟩ Referenced in 10b.
⟨ install the timbl utility 14d ⟩ Referenced in 10b.
⟨ install the tokenizer 17a ⟩ Referenced in 11a.
⟨ install the treetagger utility 12g, 13abcd, 14ab ⟩ Referenced in 10b.
⟨ install the WSD module 21a ⟩ Referenced in 11a.
⟨ install the NED module 23a ⟩ Referenced in 11a.
⟨ logmess 30a ⟩ Referenced in 8a, 9bd, 12a, 15a, 30b.
⟨ make scripts executable 11c, 17c, 18b, 19bh, 20e, 22b, 23g, 25ac, 26g, 28b, 29ac ⟩ Referenced in 42c.
⟨ make targets 36a, 39a, 42bc ⟩ Referenced in 32a.
⟨ move module 8g ⟩ Referenced in 8a, 9d, 12a, 30b.
⟨ parameters in Makefile 32de, 34bc, 36d, 39b, 42a ⟩ Referenced in 32a.
⟨ perform the task of w2html 40d ⟩ Referenced in 40c.
⟨ put spotlight jar in the Maven repository 23e ⟩ Referenced in 23a.
⟨ re-instate old module 9b ⟩ Referenced in 8a, 9d, 12a, 30b.
⟨ remove maven 6e ⟩ Referenced in 11b.
⟨ remove old module 9a ⟩ Referenced in 8a, 9d, 12a, 30b.
⟨ remove the copy of the aux file 38a ⟩ Referenced in 37c, 40d.
⟨ run tex4ht 41d ⟩ Referenced in 41b.
⟨ run the html processors 41c ⟩ Referenced in 41b.
⟨ run the html processors until the aux file remains unchanged 41b ⟩ Referenced in 40d.
⟨ run the processors until the aux file remains unchanged 38c ⟩ Referenced in 37c.
⟨ run the three processors 38b ⟩ Referenced in 38c.
⟨ set alpinohome 12f ⟩ Referenced in 18a.
⟨ set local bin directory 4e ⟩ Referenced in 27e.
⟨ set up java 5c ⟩ Referenced in 10b.
⟨ set up java environment in scripts 5e, 6a ⟩ Referenced in 4f, 10b.
⟨ set up programming environment 4g ⟩ Referenced in 17b, 18a, 19g, 20d, 22a, 24b, 25b, 28d.
⟨ set up python 6f ⟩ Referenced in 10b.
⟨ start the spotlight server 16c ⟩ Referenced in 16d.
⟨ test whether virtualenv is present on the host 7b ⟩ Referenced in 7a.
⟨ unpack snapshots or die 10a ⟩ Referenced in 10b.
⟨ unpack the java tarball 5d ⟩ Referenced in 5c.
⟨ unpack ticcutils or timbl 15a ⟩ Referenced in 14cd.
⟨ variables of install-modules 29d ⟩ Referenced in 10b.

C.3 Variables