Standardised Dutch NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

17th February 2015
13:07 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.

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

Module	Section	Source	Commit	Script
Tokenizer	3.7.1	Github	c4d307eece4ef19aca365e3a08abd7f3324e3707	`tok`
morphosyntactic parser	3.7.2	Github	c6cabea2cc37ac3098c5927f5ec5b180ac31246f	`mor`
NERC	3.7.5	Github	8b518284eface454a4f954dfb88bea4d98b2073a	nerc
WSD	3.7.6	Github	2babeb40a81b3720274a0521ccc2a27c5eff28c9	wsd
Onto-tagger	3.7.9	snapshot		onto
Heideltime	3.7.11	Github	057c93ccc857a427145b9e2ff72fd645172d34df	heideltime
SRL	3.7.12	Github	a5e63ba512cc326274b1285cf2af81ff8a2e04b5	srl
NED	3.7.8	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	3.7.4	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	3.7.13	snapshot		evcoref
Framenet SRL	3.7.10	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;* ***commit:*** *Commit-name or version-tag* ***script:*** *Script to be included in a pipeline.*

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.

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	3.6.1	RUG
Ticcutils	3.6.3	ILK
Timbl	3.6.3	ILK
Treetagger	3.6.2	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* 3 ⟩ ≡

```
../modules ◇
```

Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *directories to create* 4a ⟩ ≡
```
../bin ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *directories to create* 4b ⟩ ≡
```
../modules/usrlocal ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *directories to create* 4c ⟩ ≡
```
../modules/usrlocal/bin ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *directories to create* 4d ⟩ ≡
```
../modules/usrlocal/lib ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *directories to create* 4e ⟩ ≡
```
../modules/python ../env/java ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

Make binaries findable:

⟨ *set local bin directory* 4f ⟩ ≡
```
export PATH=/home/paul/projecten/cltl/pipelines/nlpp/modules/usrlocal/bin:$PATH
◇
```
Fragment referenced in 27a.

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" 4g≡
```
PIPEROOT=/home/paul/projecten/cltl/pipelines/nlpp
PIPEBIN=$PIPEROOT/bin
PIPEMODD=$PIPEROOT/modules
⟨ set up java environment in scripts 5f, ... ⟩
⟨ activate the python environment 7d, ... ⟩
◇
```

⟨ *set up programming environment* 5a ⟩ ≡
```
source /home/paul/projecten/cltl/pipelines/nlpp/bin/progenv
```
◇
Fragment referenced in 17ac, 19a, 20a, 21b, 23a, 24, 25a, 28b.

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/nlpp/env`.

⟨ *directories to create* 5b ⟩ ≡
```
../env/java
```
◇
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *check this first* 5c ⟩ ≡
```
if
  [ ! -e /home/paul/projecten/cltl/pipelines/nlpp/server-jre-7u72-linux-
x64.tar.gz ]
then
  echo "Cannot find  /home/paul/projecten/cltl/pipelines/nlpp/server-jre-7u72-
linux-x64.tar.gz"
  exit 4
fi
```
◇
Fragment defined by 5c, 11a.
Fragment never referenced.

⟨ *set up java* 5d ⟩ ≡
```
⟨ unpack the java tarball 5e ⟩
```
◇
Fragment never referenced.

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

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

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 6a ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *set up java environment in scripts* 6b ⟩ ≡
```
export JARDIR=/home/paul/projecten/cltl/pipelines/nlpp/env/java/jars
◇
```
Fragment defined by 5f, 6b.
Fragment referenced in 4g, 10b.

2.2 Maven

⟨ *directories to create* 6c ⟩ ≡
```
/home/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5 ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

⟨ *install maven* 6d ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/nlpp/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 6de.
Fragment never referenced.

⟨ *install maven* 6e ⟩ ≡
```
export MAVEN_HOME=/home/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```
Fragment defined by 6de.
Fragment never referenced.

⟨ *remove maven* 6f ⟩ ≡
```
rm -rf /home/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5
◇
```
Fragment referenced in 10d.

2.3 Python

⟨ *set up python* 7a ⟩ ≡
⟨ *create a virtual environment for Python* 7b ⟩
⟨ *activate the python environment* 7d, ... ⟩
⟨ *install kafnafparserpy* 8b ⟩
⟨ *install python packages* 8g ⟩
◇
Fragment referenced in 10b.

2.3.1 Virtual environment

Create a virtual environment.

⟨ *create a virtual environment for Python* 7b ⟩ ≡
⟨ *test whether virtualenv is present on the host* 7c ⟩
```
cd /home/paul/projecten/cltl/pipelines/nlpp/env
virtualenv venv
```
◇
Fragment referenced in 7a.
Uses: `virtualenv` 7c.

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

⟨ *activate the python environment* 7d ⟩ ≡
```
source /home/paul/projecten/cltl/pipelines/nlpp/env/venv/bin/activate
```
◇
Fragment defined by 7d, 8a.
Fragment referenced in 4g, 7a, 26b, 27a.
Defines: `activate` Never used.

Subdirectory `/home/paul/projecten/cltl/pipelines/nlpp/env/python` will contain general Python packages like KafnafParserPy.

⟨ *directories to create* 7e ⟩ ≡
`/home/paul/projecten/cltl/pipelines/nlpp/env/python` ◇
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

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

⟨ *activate the python environment* 8a ⟩ ≡
```
export PYTHONPATH=/home/paul/projecten/cltl/pipelines/nlpp/env/python:$PYTHONPATH
```
◇

Fragment defined by 7d, 8a.
Fragment referenced in 4g, 7a, 26b, 27a.
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* 8b ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/nlpp/env/python
DIRN=KafNafParserPy
⟨ move module (8c $DIRN ) 9a ⟩
git clone https://github.com/cltl/KafNafParserPy.git
if
  [ $? -gt 0 ]
then
  ⟨ logmess (8d Cannot install current $DIRN version ) 29c ⟩
  ⟨ re-instate old module (8e $DIRN ) 9c ⟩
else
  ⟨ remove old module (8f $DIRN ) 9b ⟩
fi
```
◇

Fragment referenced in 7a.

2.3.3 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* 8g ⟩ ≡
```
pip install lxml
pip install pyyaml
```
◇

Fragment referenced in 7a.
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* 9a ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
```
◇

Fragment referenced in 8b, 9e, 11c, 30a.

⟨ *remove old module* 9b ⟩ ≡
```
rm -rf old.@1
```
◇

Fragment referenced in 8b, 9e, 11c, 30a.

⟨ *re-instate old module* 9c ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
```
⟨ *logmess* (9d `$MESS`) 29c ⟩

◇

Fragment referenced in 8b, 9e, 11c, 30a.

3.2 Installation from Github

The following macro can be used to install a module from github. Before issuing this macto, the following four variables must be set:

MODNAM: Name of the module.
DIRN: Name of the root directory of the module.
GITU: Github URL to clone from.
GITC: Github commit-name or version tag.

⟨ *install from github* 9e ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/nlpp/modules
```
⟨ *move module* (9f `$DIRN`) 9a ⟩
```
git clone $GITU
if
  [ $? -gt 0 ]
then
```
⟨ *logmess* (9g `Cannot install current $MODNAM version`) 29c ⟩
⟨ *re-instate old module* (9h `$DIRN`) 9c ⟩
```
else
```
⟨ *remove old module* (9i `$DIRN`) 9b ⟩
```
  cd /home/paul/projecten/cltl/pipelines/nlpp/modules/$DIRN
  git checkout $GITC
fi
```
◇

Fragment referenced in 17b, 18c, 20b, 22a, 25b, 26c.

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/nlpp/nl-pipeline_snapshots_20150127.tgz`.

⟨ *unpack snapshots or die* 10a ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/nlpp
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 never referenced.

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 29b ⟩
⟨ set up java environment in scripts 5f, ... ⟩
echo ... Python
⟨ set up python 7a ⟩
echo ... Treetagger
⟨ install the treetagger utility 12b, ... ⟩
```
◇

File defined by 10bcd.

`"../bin/install-modules"` 10c≡
```
echo Install modules
echo ... Heideltime
⟨ install the heideltime module 25b ⟩
echo Final
```
◇

File defined by 10bcd.

`"../bin/install-modules"` 10d≡
```
⟨ remove maven 6f ⟩
```
◇

File defined by 10bcd.

3.5 Check availability of resources

Text for some resources that we need and that may not be available on this host.

⟨ *check this first* 11a ⟩ ≡
⟨ *check whether mercurial is present* 11b ⟩
◇
Fragment defined by 5c, 11a.
Fragment never referenced.

⟨ *check whether mercurial is present* 11b ⟩ ≡
```
which hg
if
  [ $? -ne 0 ]
then
  echo Please install mercurial.
  exit 1
fi
```
◇
Fragment referenced in 11a.
Defines: `hg` 18c.

3.6 Install utilities and resources

3.6.1 Alpino

Install Alpino from the website of Gertjan van Noort.

Module

⟨ *install Alpino* 11c ⟩ ≡
```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/nlpp/modules
```
⟨ *move module* (11d `Alpino`) 9a ⟩
```
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* (11e `Installed Alpino`) 29c ⟩
⟨ *remove old module* (11f `Alpino`) 9b ⟩
```
else
```
⟨ *re-instate old module* (11g `Alpino`) 9c ⟩
```
fi
```
◇
Fragment never referenced.

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* 12a ⟩ ≡
```
export ALPINO_HOME=/home/paul/projecten/cltl/pipelines/nlpp/modules/Alpino
```
◇

Fragment referenced in 17c.
Defines: `ALPINO_HOME` Never used.

3.6.2 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* 12b ⟩ ≡
```
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 12bcd, 13abcd.
Fragment referenced in 10b.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 12c ⟩ ≡
```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
```
◇

Fragment defined by 12bcd, 13abcd.
Fragment referenced in 10b.

Parametersets:

⟨ *install the treetagger utility* 12d ⟩ ≡
```
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 12bcd, 13abcd.
Fragment referenced in 10b.

Download everything in the target directory:

⟨ *install the treetagger utility* 13a ⟩ ≡
```
mkdir -p /home/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR
cd /home/paul/projecten/cltl/pipelines/nlpp/modules/$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 12bcd, 13abcd.
Fragment referenced in 10b.

Run the install-script:

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

Make the treetagger utilities available for everbody.

⟨ *install the treetagger utility* 13c ⟩ ≡
```
chmod -R o+rx /home/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/bin
chmod -R o+rx /home/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/cmd
chmod -R o+r /home/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/doc
chmod -R o+rx /home/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/lib
◇
```
Fragment defined by 12bcd, 13abcd.
Fragment referenced in 10b.

Remove the tarballs:

⟨ *install the treetagger utility* 13d ⟩ ≡
```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```
Fragment defined by 12bcd, 13abcd.
Fragment referenced in 10b.

3.6.3 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* 14a ⟩ ≡
```
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 14c ⟩
◇
```
Fragment never referenced.

⟨ *install the timbl utility* 14b ⟩ ≡
```
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 14c ⟩
◇
```
Fragment never referenced.

⟨ *unpack ticcutils or timbl* 14c ⟩ ≡
```
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/nlpp/modules/usrlocal
  make
  make install
fi
cd /home/paul/projecten/cltl/pipelines/nlpp
rm -rf $ticbeldir
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (14d Installed $DIR ) 29c ⟩
else
  ⟨ logmess (14e NOT installed $DIR ) 29c ⟩
fi
◇
```
Fragment referenced in 14ab.

3.6.4 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 dbpedia 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.

So, let's do that.

⟨ *install the spotlight server* 15a ⟩ ≡

```
mkdir -p /home/paul/projecten/cltl/pipelines/nlpp/env/spotlight
cd /home/paul/projecten/cltl/pipelines/nlpp/env/spotlight
cp /home/paul/projecten/cltl/pipelines/nlpp/snapshots/spotlight/dbpedia-spotlight-
0.7-jar-with-dependencies-candidates.jar .
wget http://spotlight.sztaki.hu/downloads/nl.tar.gz
tar -xzf nl.tar.gz
wget http://spotlight.sztaki.hu/downloads/en_2+2.tar.gz
tar -xzf en_2+2.tar.gz
◇
```

Fragment defined by 15ab.
Fragment never referenced.

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

⟨ *install the spotlight server* 15b ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/nlpp/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 15ab.
Fragment never referenced.

⟨ *start the spotlight server* 16a ⟩ ≡

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

◇

Fragment referenced in 16b.

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

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

◇

Fragment referenced in 23a.

3.7 Install modules

3.7.1 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* 16c ⟩ ≡

```
tempdir=‘mktemp -d -t tok.XXXXXX‘
cd $tempdir
git clone https://github.com/ixa-ehu/ixa-pipe-tok.git
git checkout c4d307eece4ef19aca365e3a08abd7f3324e3707
cd ixa-pipe-tok
mvn clean package
mv target/ixa-pipe-tok-
1.7.0.jar /home/paul/projecten/cltl/pipelines/nlpp/env/java/jars
cd /home/paul/projecten/cltl/pipelines/nlpp
```

◇

Fragment never referenced.

Script The script runs the tokenizerscript.

```
"../bin/tok" 17a≡
        #!/bin/bash
        ⟨ set up programming environment 5a ⟩
        JARFILE=/home/paul/projecten/cltl/pipelines/nlpp/env/java/jars/ixa-pipe-tok-
        1.7.0.jar
        java -Xmx1000m  -jar $JARFILE tok -l nl --inputkaf
        ◇
```

3.7.2 Morphosyntactic parser

Module

```
⟨ install the morphosyntactic parser 17b ⟩ ≡
        MODNAM=morphsynparser
        DIRN=morphosyntactic_parser_nl
        GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
        GITC=c6cabea2cc37ac3098c5927f5ec5b180ac31246f
        ⟨ install from github 9e ⟩
        cd /home/paul/projecten/cltl/pipelines/nlpp/modules/morphosyntactic_parser_nl
        git checkout c6cabea2cc37ac3098c5927f5ec5b180ac31246f
        ◇
```

Fragment never referenced.

Script

```
"../bin/mor" 17c≡
        #!/bin/bash
        ⟨ set up programming environment 5a ⟩
        ROOT=/home/paul/projecten/cltl/pipelines/nlpp
        MODDIR=/home/paul/projecten/cltl/pipelines/nlpp/modules/morphosyntactic_parser_nl
        ⟨ set alpinohome 12a ⟩
        cat | python $MODDIR/core/morph_syn_parser.py
        ◇
```

3.7.3 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 17d ⟩ ≡
        ../modules/alpinohack ◇
```

Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.

"../modules/alpinohack/clean_hack.py" 18a≡

```
#!/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 35b.

Script

"../bin/alpinohack" 18b≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
HACKDIR=/home/paul/projecten/cltl/pipelines/nlpp/modules/alpinohack
cat | python  $HACKDIR/clean_hack.py

◇
```

3.7.4 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* 18c ⟩ ≡

```
MODNAM=coreference-base
DIRN=coreference-base
GITU=https://github.com/opener-project/coreference-base.git
GITC=bfa5aec0fa498e57fe14dd4d2c51365dd09a0757
⟨ install from github 9e ⟩
pip install --upgrade  hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
pip install --upgrade  networkx
◇
```

Fragment never referenced.
Uses: hg 11b.

Script

```
"../bin/coreference-base" 19a≡
    #!/bin/bash
    ⟨ set up programming environment 5a ⟩
    cd $PIPEMODD/coreference-base/core
    cat | python -m corefgraph.process.file --language nl --singleton --sieves NO
    ◇
```

3.7.5 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 19b ⟩ ≡
    ⟨ compile the nerc jar 19c ⟩
    ⟨ get the nerc models 19d ⟩

    cp -r /home/paul/projecten/cltl/pipelines/nlpp/snapshots/EHU-
    nerc  /home/paul/projecten/cltl/pipelines/nlpp/modules/
    ◇
```

Fragment never referenced.

```
⟨ compile the nerc jar 19c ⟩ ≡
    TEMPDIR==‘mktemp -d -t nerc.XXXXXX‘
    cd $TEMPDIR
    git clone https://github.com/ixa-ehu/ixa-pipe-nerc
    cd ixa-pipe-nerc/
    git checkout 8b518284eface454a4f954dfb88bea4d98b2073a
    mvn clean package
    mv target/ixa-pipe-nerc-
    1.3.3.jar /home/paul/projecten/cltl/pipelines/nlpp/env/java/jars/
    cd /home/paul/projecten/cltl/pipelines/nlpp/nuweb
    rm -rf $TEMPDIR
    ◇
```

Fragment referenced in 19b.
Uses: nuweb 37d.

```
⟨ get the nerc models 19d ⟩ ≡
    mkdir -p ../modules/EHU-nerc
    cp -r /home/paul/projecten/cltl/pipelines/nlpp/snapshots/EHU-nerc/nerc-
    resources ../modules/EHU-nerc
    ◇
```

Fragment referenced in 19b.

Script

"../bin/nerc" 20a≡
```
#!/bin/bash
⟨ set up programming environment 5a ⟩
MODDIR=$PIPEMODD/EHU-nerc
JAR=$JARDIR/ixa-pipe-nerc-1.3.3.jar
MODEL=nl-local-conll02-testa.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL
◇
```

3.7.6 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* 20b ⟩ ≡
```
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=2babeb40a81b3720274a0521ccc2a27c5eff28c9
⟨ install from github 9e ⟩
cd /home/paul/projecten/cltl/pipelines/nlpp/modules/svm_wsd
⟨ install svm lib 20c ⟩
⟨ download svm models 21a ⟩


◇
```
Fragment never referenced.

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

⟨ *install svm lib* 20c ⟩ ≡
```
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 20b.

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

⟨ *download svm models* 21a ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/nlpp/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 20b.

Script

"../bin/wsd" 21b≡

```
#!/bin/bash
# WSD -- wrapper for word-sense disambiguation
# 8 Jan 2014 Ruben Izquierdo
# 16 sep 2014 Paul Huygen
⟨ set up programming environment 5a ⟩
WSDDIR=$PIPEMODD/svm_wsd
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $WSDDIR/$WSDSCRIPT --naf
◇
```

2

3.7.7 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* 21c ⟩ ≡

```
cp -
r /home/paul/projecten/cltl/pipelines/nlpp/snapshots/lexicalunitconvertor /home/paul/projecten/cltl/p
◇
```

Fragment never referenced.

Script

"../bin/lu2synset" 21d≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
JAVALIBDIR=/home/paul/projecten/cltl/pipelines/nlpp/modules/lexicalunitconvertor/lib
RESOURCESDIR=/home/paul/projecten/cltl/pipelines/nlpp/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
◇
```

3.7.8 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* 22a ⟩ ≡
```
⟨ put spotlight jar in the Maven repository 22b ⟩
MODNAM=ned
DIRN=ixa-pipe-ned
GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
⟨ install from github 9e ⟩
cd /home/paul/projecten/cltl/pipelines/nlpp/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/nlpp/env/java/jars/
◇
```
Fragment never referenced.

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* 22b ⟩ ≡
```
echo Put Spotlight jar in the Maven repository.
tempdir=‘mktemp -d -t simplespot.XXXXXX‘
cd $tempdir
wget http://spotlight.sztaki.hu/downloads/dbpedia-spotlight-0.7.jar
wget http://spotlight.sztaki.hu/downloads/nl.tar.gz
tar -xzf nl.tar.gz
MVN_SPOTLIGHT_OPTIONS="-Dfile=dbpedia-spotlight-0.7.jar"
MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DgroupId=ixa"
MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DartifactId=dbpedia-spotlight"
MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -Dversion=0.7"
MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -Dpackaging=jar"
MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DgeneratePom=true"
mvn install:install-file $MVN_SPOTLIGHT_OPTIONS

cd $PROJROOT
rm -rf $tempdir
◇
```
Fragment referenced in 22a.

Script

"../bin/ned" 23a≡
```
#!/bin/bash
⟨ set up programming environment 5a ⟩
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
JARDIR=/home/paul/projecten/cltl/pipelines/nlpp/env/java/jars
⟨ check/start the spotlight server 16b ⟩
cat | java -Xmx1000m -jar $JARDIR/ixa-pipe-ned-1.1.1.jar  -p 2060 -e candidates -
i /home/paul/projecten/cltl/pipelines/nlpp/env/spotlight/wikipedia-db -n nlEn
◇
```

3.7.9 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 23b ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/nlpp/modules
tar -xzf /home/paul/projecten/cltl/pipelines/nlpp/snapshots/vua-ontotagger-
v1.0.tar.gz
chmod -R o+r /home/paul/projecten/cltl/pipelines/nlpp/modules
◇
```
Fragment never referenced.

Script

"../bin/onto" 24≡

```
#!/bin/bash
⟨ set up programming environment 5a ⟩
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
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

◇
```

3.7.10 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" 25a≡

```
#!/bin/bash
⟨ set up programming environment 5a ⟩
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
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 35b.

3.7.11 Heideltime

Module

⟨ *install the heideltime module* 25b ⟩ ≡

```
MODNAM=heideltime
DIRN=NAF-HeidelTime
GITU=git@github.com:cltl/NAF-HeidelTime.git
GITC=057c93ccc857a427145b9e2ff72fd645172d34df
⟨ install from github 9e ⟩
⟨ adapt heideltime's config.props 26a ⟩

◇
```

Fragment referenced in 10c.

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

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

Fragment referenced in 25b.
Uses: print 35b.

Script

"../bin/heideltime" 26b≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
HEIDELDIR=/home/paul/projecten/cltl/pipelines/nlpp/modules/NAF-HeidelTime
TEMPDIR=‘mktemp -t -d heideltmp.XXXXXX‘
cd $HEIDELDIR
⟨ activate the python environment 7d, ... ⟩
iconv -t utf-
8//IGNORE | python $HEIDELDIR/HeidelTime_NafKaf.py $HEIDELDIR/heideltime-
standalone/ $TEMPDIR
rm -rf $TEMPDIR
◇
```

3.7.12 Semantic Role labelling

Module

⟨ *install the srl module* 26c ⟩ ≡

```
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=a5e63ba512cc326274b1285cf2af81ff8a2e04b5
⟨ install from github 9e ⟩
◇
```

Fragment never referenced.

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" 27a≡
      #!/bin/bash
      ROOT=/home/paul/projecten/cltl/pipelines/nlpp
      SRLDIR=/home/paul/projecten/cltl/pipelines/nlpp/modules/vua-srl-nl
      TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
      cd $SRLDIR
      ⟨ set local bin directory 4f ⟩
      ⟨ activate the python environment 7d, ... ⟩
      INPUTFILE=$TEMPDIR/inputfile
      FEATUREVECTOR=$TEMPDIR/csvfile
      TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
      ◇
```

File defined by 27abcde.

Create a feature-vector.

```
"../bin/srl" 27b≡
      cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
      ◇
```

File defined by 27abcde.

Run the trained model on the feature-vector.

```
"../bin/srl" 27c≡
      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 27abcde.

Insert the SRL values into the NAF file.

```
"../bin/srl" 27d≡
      python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
      ◇
```

File defined by 27abcde.

Clean up.

```
"../bin/srl" 27e≡
      rm -rf $TEMPDIR
      ◇
```

File defined by 27abcde.

3.7.13 Event coreference

Module Install the module from the snapshot.

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

```
cd /home/paul/projecten/cltl/pipelines/nlpp/modules
tar -xzf /home/paul/projecten/cltl/pipelines/nlpp/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/nlpp/env/java/jars
◇
```

Fragment never referenced.

Script

"../bin/evcoref" 28b≡

```
#!/bin/bash
⟨ set up programming environment 5a ⟩
MODROOT=$PIPEMODD/vua-eventcoreference_v2
RESOURCESDIR=$MODROOT/resources
JARFILE=/home/paul/projecten/cltl/pipelines/nlpp/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

◇
```

4 Utilities

4.1 Test script

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

"../bin/test" 29a≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/nlpp
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 37d.

4.2 Logging

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

⟨ *variables of install-modules* 29b ⟩ ≡
```
LOGLEVEL=1
◇
```
Fragment referenced in 10b.

⟨ *logmess* 29c ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
◇
```
Fragment referenced in 8b, 9ce, 11c, 14c, 30a.

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* 30a ⟩ ≡

```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/nlpp/modules
⟨ move module (30b $DIR ) 9a ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (30c Installed $DIR ) 29c ⟩
  ⟨ remove old module (30d $DIR ) 9b ⟩
else
  ⟨ re-instate old module (30e $DIR ) 9c ⟩
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_nlpp.w`. Figure 1 shows pathways to translate it into print-

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.*

able/viewable documents and to extract the program sources. Table 3 lists the tools that are

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*

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" 31a≡

```
⟨ default target 31b ⟩

⟨ parameters in Makefile 32b, ... ⟩

⟨ impliciete make regels 34c, ... ⟩
⟨ expliciete make regels 32d, ... ⟩
⟨ make targets 35b, ... ⟩
◇
```

The default target of make is `all`.

⟨ default target 31b ⟩ ≡

```
all : ⟨ all targets 32a ⟩
.PHONY : all

◇
```

Fragment referenced in 31a.
Defines: `all` Never used, `PHONY` 35a.

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

⟨ *all targets* 32a ⟩ ≡
```
nlpp.pdf◇
```
Fragment referenced in 31b.
Uses: pdf 35b.

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* 32b ⟩ ≡
```
.SUFFIXES: .pdf .w .tex .html .aux .log .php

◇
```
Fragment defined by 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Defines: SUFFIXES Never used.
Uses: pdf 35b.

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* 32c ⟩ ≡
```
NUWEB=../bin/nuweb
◇
```
Fragment defined by 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Defines: NUWEB 32d, 36de, 37ad, 40a, 41b, 42c.
Uses: nuweb 37d.

⟨ *expliciete make regels* 32d ⟩ ≡
```
$(NUWEB): ../nuweb-1.58
        cd ../nuweb-1.58 && make nuweb
        cp ../nuweb-1.58/nuweb $(NUWEB)

◇
```
Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: NUWEB 32c, nuweb 37d.

⟨ *expliciete make regels* 32e ⟩ ≡
```
../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 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: nuweb 37d.

A.5 Pre-processing

To make usable things from the raw input `a_nlpp.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* 33a ⟩ ≡

```
m4_nlpp.w : a_nlpp.w
        gawk '{if(match($$0, "@%")) {printf("%s", substr($$0,1,RSTART-
1))} else print}' a_nlpp.w \
          | gawk '{gsub(/[\\][\$$]/, "$$");print}'  > m4_nlpp.w
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: `print` 35b.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 33b ⟩ ≡

```
nlpp.w : m4_nlpp.w inst.m4
        m4 -P m4_nlpp.w > nlpp.w
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.

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* 34a ⟩ ≡
```
FIGFILES=fileschema
```
◇
Fragment defined by 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Defines: `FIGFILES` 34b, 38c.

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* 34b ⟩ ≡
```
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 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 36a, `PDF_FIG_NAMES` 36a, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 34a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 34c ⟩ ≡
```
%.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 34c, 36a, 39a.
Fragment referenced in 31a.
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 `nlpp.bib`. To create this file, copy the auxiliary file to another file `auxfil.aux`, but replace the argument of the command `\bibdata{nlpp}` 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* 35a ⟩ ≡
```
bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

.PHONY : bibfile
```
◇
Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: PHONY 31b.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 35b ⟩ ≡
```
pdf : nlpp.pdf

print : nlpp.pdf
        lpr nlpp.pdf

view : nlpp.pdf
        evince nlpp.pdf

```
◇
Fragment defined by 35b, 38b, 42ac.
Fragment referenced in 31a.
Defines: pdf 32ab, 36a, print 18a, 25a, 26a, 33a, 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* 36a ⟩ ≡
```
%.pdf : %.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*
```
◇
Fragment defined by 34c, 36a, 39a.
Fragment referenced in 31a.
Uses: pdf 35b, PDFT_NAMES 34b, PDF_FIG_NAMES 34b.

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* 36b ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 3, 4abcde, 5b, 6ac, 7e, 17d, 36b.
Fragment referenced in 42a.
Uses: nuweb 37d.

⟨ *parameters in Makefile* 36c ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
```
◇
Fragment defined by 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Uses: nuweb 37d.

⟨ *expliciete make regels* 36d ⟩ ≡
```
$(W2PDF) : nlpp.w $(NUWEB)
        $(NUWEB) nlpp.w
```
◇
Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: NUWEB 32c.

"../nuweb/bin/w2pdf" 36e≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20150217 at 1307h: Generated by nuweb from a_nlpp.w
NUWEB=/usr/local/bin/nuweb

LATEXCOMPILER=pdflatex
⟨ filenames in nuweb compile script 37b ⟩
⟨ compile nuweb 37a ⟩
```
◇
Uses: NUWEB 32c, nuweb 37d.

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* 37a ⟩ ≡

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

⟨ run the processors until the aux file remains unchanged 38a ⟩
⟨ remove the copy of the aux file 37c ⟩
◇
```

Fragment referenced in 36e.
Uses: `NUWEB` 32c, `nuweb` 37d.

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* 37b ⟩ ≡

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

Fragment referenced in 36e.
Defines: `auxfil` 38a, 40c, 41a, `indexfil` 38a, 40c, `nufil` 37d, 40c, 41b, `oldaux` 37c, 38a, 40c, 41a, `oldindexfil` 38a, 40c, `texfil` 37d, 40c, 41b, `trunk` 37d, 40c, 41bc.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 37c ⟩ ≡

```
rm $oldaux
◇
```

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

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* 37d ⟩ ≡

```
$NUWEB $nufil
$LATEXCOMPILER $texfil
makeindex $trunk
bibtex $trunk
◇
```

Fragment referenced in 38a.
Defines: `bibtex` 41bc, `makeindex` 41bc, `nuweb` 19c, 29a, 32cde, 36bce, 37a, 38b, 39e, 40a.
Uses: `nufil` 37b, 40c, `NUWEB` 32c, `texfil` 37b, 40c, `trunk` 37b, 40c.

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* 38a ⟩ ≡

```
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 37d ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
◇
```

Fragment referenced in 37a.
Uses: `auxfil` 37b, 40c, `indexfil` 37b, `oldaux` 37b, 40c, `oldindexfil` 37b.

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* 38b ⟩ ≡

```
html : ../nuweb/html/nlpp.html

◇
```

Fragment defined by 35b, 38b, 42ac.
Fragment referenced in 31a.
Uses: `nuweb` 37d.

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

Make lists of the graphics files and copy them.

⟨ *parameters in Makefile* 38c ⟩ ≡

```
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 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Uses: `FIGFILES` 34a.

⟨ *impliciete make regels* 39a ⟩ ≡

```
m4_htmldocdir/%.pstex : %.pstex
        cp  $< $@

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

◇

Fragment defined by 34c, 36a, 39a.
Fragment referenced in 31a.

Copy the nuweb file into the html directory.

⟨ *expliciete make regels* 39b ⟩ ≡

```
m4_htmlsource : nlpp.w
        cp  nlpp.w m4_htmlsource
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.

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* 39c ⟩ ≡

```
m4_4htfildest : m4_4htfilsource
        cp m4_4htfilsource m4_4htfildest
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.

Copy the bibliography.

⟨ *expliciete make regels* 39d ⟩ ≡

```
m4_htmlbibfil : m4_nuwebdir/nlpp.bib
        cp m4_nuwebdir/nlpp.bib m4_htmlbibfil
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.

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

⟨ *expliciete make regels* 39e ⟩ ≡

```
../nuweb/html/nlpp.html : m4_htmlsource m4_4htfildest $(HTML_PS_FIG_NAMES) $(HTML_PST_NAMES) m4_htmlb
        cp w2html ../bin
        cd ../bin && chmod 775 w2html
        cd m4_htmldocdir && ../bin/w2html nlpp.w
```

◇

Fragment defined by 32de, 33ab, 35a, 36d, 39bcde.
Fragment referenced in 31a.
Uses: `nuweb` 37d.

Create a script that performs the translation.

"w2html" 40a≡
```
#!/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 40c ⟩

⟨ perform the task of w2html 40b ⟩

◇
```
Uses: NUWEB 32c, nuweb 37d.

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 40b ⟩ ≡
```
⟨ run the html processors until the aux file remains unchanged 41a ⟩
⟨ remove the copy of the aux file 37c ⟩
◇
```
Fragment referenced in 40a.

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

⟨ *run the html processors until the aux file remains unchanged* 41a ⟩ ≡

```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 41b ⟩
done
⟨ run tex4ht 41c ⟩

◇
```

Fragment referenced in 40b.
Uses: `auxfil` 37b, 40c, `oldaux` 37b, 40c.

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

⟨ *run the html processors* 41b ⟩ ≡

```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```

Fragment referenced in 41a.
Uses: `bibtex` 37d, `makeindex` 37d, `nufil` 37b, 40c, `NUWEB` 32c, `texfil` 37b, 40c, `trunk` 37b, 40c.

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* 41c ⟩ ≡

```
tex '\def\filename{{nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
◇
```

Fragment referenced in 41a.
Uses: `bibtex` 37d, `makeindex` 37d, `trunk` 37b, 40c.

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* 41d ⟩ ≡

```
MKDIR = mkdir -p

◇
```

Fragment defined by 32bc, 34ab, 36c, 38c, 41d.
Fragment referenced in 31a.
Defines: `MKDIR` 42a.

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

$(DIRS) :
        $(MKDIR) $@
```
◇
Fragment defined by 35b, 38b, 42ac.
Fragment referenced in 31a.
Defines: DIRS 42c.
Uses: MKDIR 41d.

⟨ *make scripts executable* 42b ⟩ ≡
```
chmod -R 775  ../bin/*
```
◇
Fragment referenced in 42c.

⟨ *make targets* 42c ⟩ ≡
```
sources : nlpp.w $(DIRS) $(NUWEB)
        $(NUWEB) nlpp.w
        ⟨ make scripts executable 42b ⟩
```
◇
Fragment defined by 35b, 38b, 42ac.
Fragment referenced in 31a.
Uses: DIRS 42a, NUWEB 32c.

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 srl module 26c ⟩ Not referenced.
⟨ install the ticcutils utility 14a ⟩ Not referenced.
⟨ install the timbl utility 14b ⟩ Not referenced.
⟨ install the tokenizer 16c ⟩ Not referenced.
⟨ install the treetagger utility 12bcd, 13abcd ⟩ Referenced in 10b.
⟨ install the WSD module 20b ⟩ Not referenced.
⟨ install the NED module 22a ⟩ Not referenced.
⟨ logmess 29c ⟩ Referenced in 8b, 9ce, 11c, 14c, 30a.
⟨ make scripts executable 42b ⟩ Referenced in 42c.
⟨ make targets 35b, 38b, 42ac ⟩ Referenced in 31a.
⟨ move module 9a ⟩ Referenced in 8b, 9e, 11c, 30a.
⟨ parameters in Makefile 32bc, 34ab, 36c, 38c, 41d ⟩ Referenced in 31a.
⟨ perform the task of w2html 40b ⟩ Referenced in 40a.
⟨ put spotlight jar in the Maven repository 22b ⟩ Referenced in 22a.
⟨ re-instate old module 9c ⟩ Referenced in 8b, 9e, 11c, 30a.
⟨ remove maven 6f ⟩ Referenced in 10d.
⟨ remove old module 9b ⟩ Referenced in 8b, 9e, 11c, 30a.
⟨ remove the copy of the aux file 37c ⟩ Referenced in 37a, 40b.
⟨ run tex4ht 41c ⟩ Referenced in 41a.
⟨ run the html processors 41b ⟩ Referenced in 41a.
⟨ run the html processors until the aux file remains unchanged 41a ⟩ Referenced in 40b.
⟨ run the processors until the aux file remains unchanged 38a ⟩ Referenced in 37a.
⟨ run the three processors 37d ⟩ Referenced in 38a.
⟨ set alpinohome 12a ⟩ Referenced in 17c.
⟨ set local bin directory 4f ⟩ Referenced in 27a.
⟨ set up java 5d ⟩ Not referenced.
⟨ set up java environment in scripts 5f, 6b ⟩ Referenced in 4g, 10b.
⟨ set up programming environment 5a ⟩ Referenced in 17ac, 19a, 20a, 21b, 23a, 24, 25a, 28b.
⟨ set up python 7a ⟩ Referenced in 10b.
⟨ start the spotlight server 16a ⟩ Referenced in 16b.
⟨ test whether virtualenv is present on the host 7c ⟩ Referenced in 7b.
⟨ unpack snapshots or die 10a ⟩ Not referenced.
⟨ unpack the java tarball 5e ⟩ Referenced in 5d.
⟨ unpack ticcutils or timbl 14c ⟩ Referenced in 14ab.
⟨ variables of install-modules 29b ⟩ Referenced in 10b.

C.3 Variables