Standardised Dutch NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

23rd March 2015
10:45 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	3.7.4	Github	9927fdb32d943f0aa9748a656958af99eeb1f5b7	nerc
WSD	3.7.5	Github	2babeb40a81b3720274a0521ccc2a27c5eff28c9	wsd
Onto-tagger	3.7.8	snapshot		onto
Heideltime	3.7.10	Github	057c93ccc857a427145b9e2ff72fd645172d34df	heideltime
SRL	3.7.11	Github	675d22d361289ede23df11dcdb17195f008c54bf	srl
NED	3.7.7	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	3.7.3	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	3.7.12	snapshot		evcoref
Framenet SRL	3.7.9	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. In table 1 this has been indicated as SNAPSHOT.

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

Module	**Version**	**Section**	**Source**
KafNafParserPy	Feb 1, 2015	2.3.3	Github
Alpino	20706	3.6.1	RUG
Ticcutils	0.7	3.6.3	ILK
Timbl	6.4.6	3.6.3	ILK
Treetagger	3.2	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.

env: For the programming environment. Contains the Python local environment, the Java development kit/runtime, a directory `jars` for jars and and a directory `bin` for binaries.

modules: Contains the program code of each module in a subdirectory.

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* ? ⟩ ≡
```
    ../modules ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *directories to create* ? ⟩ ≡
```
../bin ../env/usrlocal/bin◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *directories to create* ? ⟩ ≡
```
../env/usrlocal/bin ../env/usrlocal/lib ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *directories to create* ? ⟩ ≡
```
../modules/python ../env/java ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Communicate the file-structure to scripts with a “source” script that sets variables.

`"../bin/progenv"` ?≡
```
PIPEROOT=/Users/paul/projecten/cltl/pipelines/nlpp
PIPEBIN=$PIPEROOT/bin
PIPEMODD=$PIPEROOT/modules
export PATH=/Users/paul/projecten/cltl/pipelines/nlpp/env/usrlocal/bin:$PATH
◇
```
File defined by ?, ?.

Make binaries findable:

⟨ *set local bin directory* ? ⟩ ≡
```
export PATH=/Users/paul/projecten/cltl/pipelines/nlpp/env/usrlocal/bin:$PATH
◇
```
Fragment referenced in ?.

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"` ?≡
```
⟨ set up java environment in scripts ?, ... ⟩
⟨ activate the python environment ?, ... ⟩
◇
```
File defined by ?, ?.

⟨ *set up programming environment* ? ⟩ ≡
```
source /Users/paul/projecten/cltl/pipelines/nlpp/bin/progenv
◇
```
Fragment referenced in ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?.

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 `/Users/paul/projecten/cltl/pipelines/nlpp/env`.

⟨ *directories to create* ? ⟩ ≡
```
../env/java ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *check this first* ? ⟩ ≡
```
if
  [ ! -e /Users/paul/projecten/cltl/pipelines/nlpp/server-jre-7u72-linux-
x64.tar.gz ]
then
  echo "Cannot find  /Users/paul/projecten/cltl/pipelines/nlpp/server-jre-7u72-
linux-x64.tar.gz"
  exit 4
fi
◇
```
Fragment defined by ?, ?.
Fragment referenced in ?.

⟨ *set up java* ? ⟩ ≡
 ⟨ *unpack the java tarball* ? ⟩
 ◇

Fragment referenced in ?.

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

⟨ *set up java environment in scripts* ? ⟩ ≡
```
export JAVA_HOME=/Users/paul/projecten/cltl/pipelines/nlpp/env/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by ?, ?.
Fragment referenced in ?, ?.
Defines: `JAVA_HOME` Never used.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create ?* ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *set up java environment in scripts ?* ⟩ ≡
```
export JARDIR=/Users/paul/projecten/cltl/pipelines/nlpp/env/java/jars
◇
```
Fragment defined by ?, ?.
Fragment referenced in ?, ?.

2.2 Maven

⟨ *directories to create ?* ⟩ ≡
```
/Users/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5 ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

⟨ *install maven ?* ⟩ ≡
```
cd /Users/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 ?, ?.
Fragment referenced in ?.

⟨ *install maven ?* ⟩ ≡
```
export MAVEN_HOME=/Users/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```
Fragment defined by ?, ?.
Fragment referenced in ?.

When the installation has been done, remove maven, because it is no longer needed.

⟨ *remove maven ?* ⟩ ≡
```
rm -rf /Users/paul/projecten/cltl/pipelines/nlpp/env/apache-maven-3.0.5
◇
```
Fragment referenced in ?.

2.3 Python

Set up the environment for Python. I could not find an easy way to set up Python from scratch, so we have to rely on Python 2.7 being available on the host. However, we can make a virtual environment, so that we are not dependent on the existence of libraries in the right version on the host.

In the virtual environment we will install KafNafParserPy and other Python packages that are needed.

⟨ *set up python* ? ⟩ ≡
```
⟨ check/install the correct version of python ? ⟩
⟨ create a virtual environment for Python ? ⟩
⟨ activate the python environment ?, ... ⟩
⟨ install kafnafparserpy ? ⟩
⟨ install python packages ? ⟩
◇
```
Fragment referenced in ?.

2.3.1 Python version

The pipeline relies on Python version 2.7 being available. If possible, the user should provide this version and make sure that the "python" command invokes version 2.7.something of python. However, ikn some cases (notably in the case of a Centos 6.3 server) this is difficult to achieve. In that case we can use a binary python supplied by ActivePython (http://www.activestate.com/activepython). Download in that case the tarball `ActivePython-2.7.8.10-linux-x86_64.tar.gz` from the ActivePython site and put it in the `nlpp` directory. The following macro checks whether the `python` command invokes a correct version of python and, if this is not the case and the ActivePython tarball is present, install ActivePython.

⟨ *check/install the correct version of python* ? ⟩ ≡
```
pythonok=`python --
version 2>&1 | gawk '{if(match($2, "2.7")) print "yes" ; else print "no" }'`
if
  [ "$pythonok" == "no" ]
then
  ⟨ install ActivePython ?, ... ⟩
fi
◇
```
Fragment referenced in ?.
Defines: `pythonok` Never used.
Uses: `print` ?.

Check whether we have the ActivePython tarball and quit if tis is not the case.

⟨ *install ActivePython* ? ⟩ ≡
```
actpyt=`ls -1 /Users/paul/projecten/cltl/pipelines/nlpp/ActivePython*gz`
if
  [ $? -gt 0 ]
then
  echo "Cannot install Python 2.7."
  echo "Please put ActivePython tarball in nlpp directory."
  exit 1
fi
◇
```
Fragment defined by ?, ?.
Fragment referenced in ?.

Unpack the tarball in a temporary directory and install active python in the `env` subdirectory of nlpp. It turns out that you must upgrade pip, virtualenv and setuptools after the installation (see https://github.com/ActiveState/activepython-docker/commit/10fff72069e51dbd36330cb8a7c2f0845bcd7b3 and https://github.com/ActiveState/activepython-docker/issues/1).

⟨ *install ActivePython ?* ⟩ ≡

```
pytinsdir=‘mktemp -d -t activepyt.XXXXXX‘
cd $pytinsdir
tar -xzf $actpyt
acdir=‘ls -1‘
cd $acdir
./install.sh -I /Users/paul/projecten/cltl/pipelines/nlpp/env/usrlocal
cd /Users/paul/projecten/cltl/pipelines/nlpp
rm -rf $pytinsdir
pip install -U pip virtualenv setuptools
\◇
```

Fragment defined by ?, ?.
Fragment referenced in ?.
Uses: `virtualenv` ?.

2.3.2 Virtual environment

Create a virtual environment. To begin this, we need the python module virtualenv on the host.

⟨ *create a virtual environment for Python ?* ⟩ ≡

```
⟨ test whether virtualenv is present on the host ? ⟩
cd /Users/paul/projecten/cltl/pipelines/nlpp/env
virtualenv venv
◇
```

Fragment referenced in ?.
Uses: `virtualenv` ?.

⟨ *test whether virtualenv is present on the host ?* ⟩ ≡

```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
◇
```

Fragment referenced in ?.
Defines: `virtualenv` ?, ?.

⟨ *activate the python environment ?* ⟩ ≡

```
source /Users/paul/projecten/cltl/pipelines/nlpp/env/venv/bin/activate
◇
```

Fragment defined by ?, ?.
Fragment referenced in ?, ?.
Defines: `activate` Never used.

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

⟨ *directories to create* ? ⟩ ≡
```
        /Users/paul/projecten/cltl/pipelines/nlpp/env/python ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

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

⟨ *activate the python environment* ? ⟩ ≡
```
        export PYTHONPATH=/Users/paul/projecten/cltl/pipelines/nlpp/env/python:$PYTHONPATH
        ◇
```
Fragment defined by ?, ?.
Fragment referenced in ?, ?.
Defines: `PYTHONPATH` Never used.

2.3.3 KafNafParserPy

A cornerstone Pythonmodule for the pipeline is KafNafParserPy. It is a feature of this module that you cannot install it with PIP, but that you can add it to your PYTHONPATH.

⟨ *install kafnafparserpy* ? ⟩ ≡
```
        cd /Users/paul/projecten/cltl/pipelines/nlpp/env/python
        DIRN=KafNafParserPy
        ⟨ move module (? $DIRN ) ? ⟩
        git clone https://github.com/cltl/KafNafParserPy.git
        if
          [ $? -gt 0 ]
        then
          ⟨ logmess (? Cannot install current $DIRN version ) ? ⟩
          ⟨ re-instate old module (? $DIRN ) ? ⟩
        else
          ⟨ remove old module (? $DIRN ) ? ⟩
        fi
        ◇
```
Fragment referenced in ?.

2.3.4 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* ? ⟩ ≡
```
        pip install lxml
        pip install pyyaml
        ◇
```
Fragment referenced in ?.
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* ? ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
◇
```
Fragment referenced in ?, ?, ?, ?.

⟨ *remove old module* ? ⟩ ≡
```
rm -rf old.@1
◇
```
Fragment referenced in ?, ?, ?, ?.

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

◇
```
Fragment referenced in ?, ?, ?, ?.

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* ? ⟩ ≡
```
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules
⟨ move module (? $DIRN ) ? ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (? Cannot install current $MODNAM version ) ? ⟩
  ⟨ re-instate old module (? $DIRN ) ? ⟩
else
  ⟨ remove old module (? $DIRN ) ? ⟩
  cd /Users/paul/projecten/cltl/pipelines/nlpp/modules/$DIRN
  git checkout $GITC
fi

◇
```
Fragment referenced in ?, ?, ?, ?, ?, ?.

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 `/Users/paul/projecten/cltl/pipelines/nlpp/nl-pipeline_snapshots_20150309.tgz`.

⟨ *unpack snapshots or die* ? ⟩ ≡
```
cd /Users/paul/projecten/cltl/pipelines/nlpp
if
  [ -e nl-pipeline_snapshots_20150309.tgz ]
then
  tar -zxf nl-pipeline_snapshots_20150309.tgz
fi
if
  [ ! -e snapshots ]
then
  echo "No module snapshots"
  exit 1
fi
◇
```
Fragment referenced in ?.

3.4 The installation script

The installation is performed by script `install-modules`

```
"../bin/install-modules" ?≡
        #!/bin/bash
        echo Set up environment
        ⟨ set local bin directory ? ⟩
        ⟨ variables of install-modules ? ⟩
        ⟨ check this first ?, ... ⟩
        ⟨ unpack snapshots or die ? ⟩
        echo ... Java
        ⟨ set up java ? ⟩
        ⟨ set up java environment in scripts ?, ... ⟩
        ⟨ install maven ?, ... ⟩
        echo ... Python
        ⟨ set up python ? ⟩
        echo ... Alpino
        ⟨ install Alpino ? ⟩
        echo ... Spotlight
        ⟨ install the Spotlight server ?, ... ⟩
        echo ... Treetagger
        ⟨ install the treetagger utility ?, ... ⟩
        echo ... Ticcutils and Timbl
        ⟨ install the ticcutils utility ? ⟩
        ⟨ install the timbl utility ? ⟩
        ◇
```
File defined by ?, ?, ?.

```
"../bin/install-modules" ?≡
        echo Install modules
        echo ... Tokenizer
        ⟨ install the tokenizer ? ⟩
        echo ... Morphosyntactic parser
        ⟨ install the morphosyntactic parser ? ⟩
        echo ... NERC
        ⟨ install the NERC module ? ⟩
        echo ... Coreference base
        ⟨ install coreference-base ? ⟩
        echo ... WSD
        ⟨ install the WSD module ? ⟩
        echo ... Ontotagger
        ⟨ install the onto module ? ⟩
        echo ... Heideltime
        ⟨ install the heideltime module ? ⟩
        echo ... SRL
        ⟨ install the srl module ? ⟩
        echo ... NED
        ⟨ install the NED module ? ⟩
        echo ... Event-coreference
        ⟨ install the event-coreference module ? ⟩
        echo ... lu2synset
        ⟨ install the lu2synset converter ? ⟩
        echo Final
        ◇
```
File defined by ?, ?, ?.

"../bin/install-modules" ?≡
    ⟨ *remove maven* ? ⟩
    ◇

File defined by ?, ?, ?.

3.5 Check availability of resources

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

⟨ *check this first* ? ⟩ ≡
    ⟨ *check whether mercurial is present* ? ⟩
    ◇

Fragment defined by ?, ?.
Fragment referenced in ?.

⟨ *check whether mercurial is present* ? ⟩ ≡

```
which hg
if
  [ $? -ne 0 ]
then
  echo Please install Mercurial.
  exit 1
fi
```

◇

Fragment referenced in ?.
Defines: `hg` ?.

3.6 Install utilities and resources

3.6.1 Alpino

Install Alpino from the website of Gertjan van Noort.

Module

⟨ *install Alpino* ? ⟩ ≡
```
SUCCES=0
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules
⟨ move module (? Alpino ) ? ⟩
wget http://www.let.rug.nl/vannoord/alp/Alpino/binary/versions/Alpino-x86_64-linux-
glibc2.5-20706-sicstus.tar.gz
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
  SUCCES=$?
  rm -rf Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (? Installed Alpino ) ? ⟩
  ⟨ remove old module (? Alpino ) ? ⟩
else
  ⟨ re-instate old module (? Alpino ) ? ⟩
fi
◇
```
Fragment referenced in ?.

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* ? ⟩ ≡
```
export ALPINO_HOME=/Users/paul/projecten/cltl/pipelines/nlpp/modules/Alpino
◇
```
Fragment referenced in ?.
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 Treetagger comes from and the location where it is going to reside:

⟨ *install the treetagger utility* ? ⟩ ≡
```
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 ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* ? ⟩ ≡
```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Parametersets:

⟨ *install the treetagger utility* ? ⟩ ≡
```
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 ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Download everything in the target directory:

⟨ *install the treetagger utility* ? ⟩ ≡
```
mkdir -p /Users/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules/$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 ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Run the install-script:

⟨ *install the treetagger utility* ? ⟩ ≡
```
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Make the treetagger utilities available for everbody.

⟨ *install the treetagger utility* ? ⟩ ≡
```
chmod -R o+rx /Users/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/bin
chmod -R o+rx /Users/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/cmd
chmod -R o+r /Users/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/doc
chmod -R o+rx /Users/paul/projecten/cltl/pipelines/nlpp/modules/$TREETAGDIR/lib
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Remove the tarballs:

⟨ *install the treetagger utility* ? ⟩ ≡
```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

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* ? ⟩ ≡
```
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 ? ⟩
◇
```
Fragment referenced in ?.

⟨ *install the timbl utility* ? ⟩ ≡
```
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 ? ⟩
◇
```
Fragment referenced in ?.

⟨ *unpack ticcutils or timbl* ? ⟩ ≡
```
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=/Users/paul/projecten/cltl/pipelines/nlpp/env/usrlocal
  make
  make install
fi
cd /Users/paul/projecten/cltl/pipelines/nlpp
rm -rf $ticbeldir
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (? Installed $DIR ) ?⟩
else
  ⟨ logmess (? NOT installed $DIR ) ?⟩
fi
◇
```
Fragment referenced in ?, ?.

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* ? ⟩ ≡
```
mkdir -p /Users/paul/projecten/cltl/pipelines/nlpp/env/spotlight
cd /Users/paul/projecten/cltl/pipelines/nlpp/env/spotlight
cp /Users/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 ?, ?.
Fragment referenced in ?.

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

⟨ *install the Spotlight server* ? ⟩ ≡
```
cd /Users/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 ?, ?.
Fragment referenced in ?.

⟨ *start the Spotlight server* ? ⟩ ≡
```
cd /Users/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 ?.

⟨ *check/start the Spotlight server* ? ⟩ ≡
```
spottasks=‘netstat -an | grep :2060 | wc -l‘
if
  [ $spottasks -eq 0 ]
then
  ⟨ start the Spotlight server ? ⟩
  sleep 60
fi
◇
```
Fragment referenced in ?.

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 ?* ⟩ ≡

```
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 /Users/paul/projecten/cltl/pipelines/nlpp/env/java/jars
cd /Users/paul/projecten/cltl/pipelines/nlpp
rm -rf $tempdir
◇
```

Fragment referenced in ?.

Script The script runs the tokenizerscript.

"../bin/tok" ?≡

```
#!/bin/bash
⟨ set up programming environment ? ⟩
JARFILE=/Users/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 ?* ⟩ ≡

```
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=c6cabea2cc37ac3098c5927f5ec5b180ac31246f
⟨ install from github ? ⟩
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules/morphosyntactic_parser_nl
git checkout c6cabea2cc37ac3098c5927f5ec5b180ac31246f
◇
```

Fragment referenced in ?.

Script

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

3.7.3 Nominal coreference-base

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

Module

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

Fragment referenced in ?.
Uses: hg ?.

Script

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

3.7.4 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, models have been put in the snapshot-tarball.

```
⟨ install the NERC module ? ⟩ ≡
        ⟨ compile the nerc jar ? ⟩
        ⟨ get the nerc models ? ⟩

        ◇
```

Fragment referenced in ?.

The nerc module is a Java program that is contained in a jar. Pul the source from Github in a temporary directory, compile the jar with java and move the jar to the jars directory.

⟨ *compile the nerc jar* ? ⟩ ≡

```
TEMPDIR==`mktemp -d -t nerc.XXXXXX`
cd $TEMPDIR
git clone https://github.com/ixa-ehu/ixa-pipe-nerc
cd ixa-pipe-nerc/
git checkout 9927fdb32d943f0aa9748a656958af99eeb1f5b7
mvn clean package
mv target/ixa-pipe-nerc-
1.3.6.jar /Users/paul/projecten/cltl/pipelines/nlpp/env/java/jars/
cd /Users/paul/projecten/cltl/pipelines/nlpp/nuweb
rm -rf $TEMPDIR
◇
```

Fragment referenced in ?.
Uses: `nuweb` ?.

The current version of the pipeline uses the following models, that have been made avaiable by Rodrigo Agerri on march 2, 2015. Rodrigo wrote:

```
  I have recently trained new models for Dutch using both the CoNLL 2002
and the Sonar corpora. These models are better than the one currently
being used in the Dutch Newsreader pipeline. They are not yet in the
resources of the ixa pipes (no public yet) but in the meantime they
might be useful if you plan to do some processing in Dutch.


For CoNLL 2002, the new model obtains 83.46 F1, being the previously
best published result 77.05 on that dataset.
The Sonar model is trained on the full corpus, and evaluated using
random 10 fold cross validation. The only previous result I know of
obtains 80.71 F1 wrt to our model which obtains 87.84. However,
because it is not evaluated on a separate test partition I do not take
these results too seriously.


You will need to update the ixa-pipe-nerc module. The CoNLL 2002 model
runs as before but to use the Sonar model you need to add the extra
parameter --clearFeatures yes, like this:


Sonar model: cat file.pos.naf | java -jar ixa-pipe-nerc-1.3.6.jar tag
-m $nermodel --clearFeatures yes
CoNLL model: cat file.pos.naf | java -jar ixa-pipe-nerc-1.3.6.jar tag
-m $nermodel

http://www.lt3.ugent.be/en/publications/fine-grained-dutch-named-entity-recognition/

[..]
In any case, here are the models.

http://ixa2.si.ehu.es/ragerri/dutch-nerc-models.tar.gz
```

The tarball `dutch-nerc-models.tar.gz` contains the models `nl-clusters-conll02.bin` and `nl-clusters-sonar.bin` Both models have been placed in subdirectory `/EHU-nerc/nerc-resources/nl` of the snapshot.

⟨ *get the nerc models* ? ⟩ ≡
```
mkdir -p /Users/paul/projecten/cltl/pipelines/nlpp/modules/EHU-nerc
cp -r /Users/paul/projecten/cltl/pipelines/nlpp/snapshots/EHU-nerc/nerc-
resources /Users/paul/projecten/cltl/pipelines/nlpp/modules/EHU-nerc/
chmod -R 775 /Users/paul/projecten/cltl/pipelines/nlpp/modules/EHU-nerc
◇
```
Fragment referenced in ?.

Script Make a script that uses the conll02 model and a script that uses the Sonar model

"../bin/nerc_conll02" ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
MODDIR=$PIPEMODD/EHU-nerc
JAR=$JARDIR/ixa-pipe-nerc-1.3.6.jar
MODEL=nl-clusters-conll02.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL
#cat| java           -jar ixa-pipe-nerc-1.3.6.jar tag -m $nermodel
◇
```

"../bin/nerc_sonar" ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
MODDIR=$PIPEMODD/EHU-nerc
JAR=$JARDIR/ixa-pipe-nerc-1.3.6.jar
MODEL=nl-clusters-sonar.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL --
clearFeatures yes
#cat| java           -jar ixa-pipe-nerc-1.3.6.jar tag -m $nermodel --
clearFeatures yes
◇
```

3.7.5 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* ? ⟩ ≡
```
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=2babeb40a81b3720274a0521ccc2a27c5eff28c9
⟨ install from github ? ⟩
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules/svm_wsd
⟨ install svm lib ? ⟩
⟨ download svm models ? ⟩



◇
```
Fragment referenced in ?.

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

⟨ *install svm lib* ? ⟩ ≡
```
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 ?.

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

⟨ *download svm models* ? ⟩ ≡
```
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules/svm_wsd
cp -r /Users/paul/projecten/cltl/pipelines/nlpp/snapshots/svm_wsd/models .

◇
```
Fragment referenced in ?.

Script

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

3.7.6 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 ?* ⟩ ≡
```
cp -
r /Users/paul/projecten/cltl/pipelines/nlpp/snapshots/lexicalunitconvertor /Users/paul/projecten/cltl
◇
```
Fragment referenced in ?.

Script

"../bin/lu2synset" ?≡
```
#!/bin/bash
ROOT=/Users/paul/projecten/cltl/pipelines/nlpp
JAVALIBDIR=/Users/paul/projecten/cltl/pipelines/nlpp/modules/lexicalunitconvertor/lib
RESOURCESDIR=/Users/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
◇
```

3.7.7 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 ?* ⟩ ≡
```
⟨ put spotlight jar in the Maven repository ? ⟩
MODNAM=ned
DIRN=ixa-pipe-ned
GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
⟨ install from github ? ⟩
cd /Users/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 /Users/paul/projecten/cltl/pipelines/nlpp/env/java/jars/
◇
```
Fragment referenced in ?.

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* ? ⟩ ≡
```
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 ?.

Script

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

3.7.8 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* ? ⟩ ≡
```
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules
tar -xzf /Users/paul/projecten/cltl/pipelines/nlpp/snapshots/vua-ontotagger-
v1.0.tar.gz
chmod -R o+r /Users/paul/projecten/cltl/pipelines/nlpp/modules
◇
```
Fragment referenced in ?.

Script

"../bin/onto" ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
ROOT=/Users/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

◇
```

3.7.9 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" ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
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` ?.

3.7.10 Heideltime

Module

⟨ *install the heideltime module* ? ⟩ ≡
```
MODNAM=heideltime
DIRN=NAF-HeidelTime
GITU=https://github.com/cltl/NAF-HeidelTime.git
GITC=057c93ccc857a427145b9e2ff72fd645172d34df
⟨ install from github ? ⟩
⟨ adapt heideltime's config.props ? ⟩

◇
```
Fragment referenced in ?.

⟨ *adapt heideltime's config.props ?* ⟩ ≡
```
CONFIL=/Users/paul/projecten/cltl/pipelines/nlpp/modules/NAF-
HeidelTime/config.props
tempfil=‘mktemp -t heideltmp.XXXXXX‘
mv $CONFIL $tempfil
MODDIR=/Users/paul/projecten/cltl/pipelines/nlpp/modules
TREETAGDIR=treetagger
AWKCOMMAND='/^treeTaggerHome/ {$0="treeTagger-
Home = /Users/paul/projecten/cltl/pipelines/nlpp/modules/treetagger"}; {print}'
gawk "$AWKCOMMAND" $tempfil >$CONFIL
rm -rf $tempfil
◇
```
Fragment referenced in ?.
Uses: `print` ?.

Script

`"../bin/heideltime"` ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
HEIDELDIR=/Users/paul/projecten/cltl/pipelines/nlpp/modules/NAF-HeidelTime
TEMPDIR=‘mktemp -t -d heideltmp.XXXXXX‘
cd $HEIDELDIR
iconv -t utf-
8//IGNORE | python $HEIDELDIR/HeidelTime_NafKaf.py $HEIDELDIR/heideltime-
standalone/ $TEMPDIR
rm -rf $TEMPDIR
◇
```

3.7.11 Semantic Role labelling

Module

⟨ *install the srl module ?* ⟩ ≡
```
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=675d22d361289ede23df11dcdb17195f008c54bf
⟨ install from github ? ⟩
◇
```
Fragment referenced in ?.

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" ?≡
```
#!/bin/bash
source /Users/paul/projecten/cltl/pipelines/nlpp/bin/progenv
ROOT=$PIPEROOT
SRLDIR=/Users/paul/projecten/cltl/pipelines/nlpp/modules/vua-srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $SRLDIR
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
◇
```
File defined by ?, ?, ?, ?, ?.

Create a feature-vector.

"../bin/srl" ?≡
```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
◇
```
File defined by ?, ?, ?, ?, ?.

Run the trained model on the feature-vector.

"../bin/srl" ?≡
```
timbl -mO:I1,2,3,4 -i 25Feb2015_e-mags_mags_press_newspapers.wgt -
t $FEATUREVECTOR -o $TIMBLOUTPUTFILE >/dev/null 2>/dev/null
◇
```
File defined by ?, ?, ?, ?, ?.

Insert the SRL values into the NAF file.

"../bin/srl" ?≡
```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
◇
```
File defined by ?, ?, ?, ?, ?.

Clean up.

"../bin/srl" ?≡
```
rm -rf $TEMPDIR
◇
```
File defined by ?, ?, ?, ?, ?.

3.7.12 Event coreference

Module Install the module from the snapshot.

⟨ *install the event-coreference module* ? ⟩ ≡
```
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules
tar -xzf /Users/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 /Users/paul/projecten/cltl/pipelines/nlpp/env/java/jars
◇
```
Fragment referenced in ?.

Script

"../bin/evcoref" ?≡
```
#!/bin/bash
⟨ set up programming environment ? ⟩
MODROOT=$PIPEMODD/vua-eventcoreference_v2
RESOURCESDIR=$MODROOT/resources
JARFILE=/Users/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" ?≡
```
#!/bin/bash
ROOT=/Users/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_conll02 > $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` ?.

4.2 Logging

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

⟨ *variables of install-modules* ? ⟩ ≡

```
LOGLEVEL=1
◇
```

Fragment referenced in ?.

⟨ *logmess* ? ⟩ ≡

```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
◇
```

Fragment referenced in ?, ?, ?, ?, ?, ?.

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* ? ⟩ ≡

```
SUCCES=0
cd /Users/paul/projecten/cltl/pipelines/nlpp/modules
⟨ move module (? $DIR ) ? ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (? Installed $DIR ) ? ⟩
  ⟨ remove old module (? $DIR ) ? ⟩
else
  ⟨ re-instate old module (? $DIR ) ? ⟩
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" ?≡
        ⟨ default target ? ⟩

        ⟨ parameters in Makefile ?, ... ⟩

        ⟨ impliciete make regels ?, ... ⟩
        ⟨ expliciete make regels ?, ... ⟩
        ⟨ make targets ?, ... ⟩
        ◇
```

The default target of make is `all`.

```
⟨ default target ? ⟩ ≡
        all : ⟨ all targets ? ⟩
        .PHONY : all

        ◇
```

Fragment referenced in ?.
Defines: `all` Never used, `PHONY` ?.

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

```
⟨ all targets ? ⟩ ≡
        nlpp.pdf◇
```

Fragment referenced in ?.
Uses: `pdf` ?.

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 ? ⟩ ≡
        .SUFFIXES: .pdf .w .tex .html .aux .log .php

        ◇
```

Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `SUFFIXES` Never used.
Uses: `pdf` ?.

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.

Put the nuweb binary in the nuweb subdirectory, so that it can be used before the directory-structure has been generated.

⟨ *parameters in Makefile* ? ⟩ ≡
```
NUWEB=./nuweb
```
◇

Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `NUWEB` ?, ?, ?, ?, ?, ?, ?, ?.
Uses: `nuweb` ?.

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

Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `NUWEB` ?, `nuweb` ?.

⟨ *expliciete make regels* ? ⟩ ≡
```
../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 ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `nuweb` ?.

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* ? ⟩ ≡
```
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 ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `print` ?.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* ? ⟩ ≡

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

◇

Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

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* ? ⟩ ≡

```
FIGFILES=fileschema
```

◇

Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `FIGFILES` ?.

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* ? ⟩ ≡

```
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 ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` ?, `PDF_FIG_NAMES` ?, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` ?.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* ? ⟩ ≡
```
%.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 ?, ?, ?.
Fragment referenced in ?.
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* ? ⟩ ≡
```
bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

.PHONY : bibfile
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `PHONY` ?.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* ? ⟩ ≡
```
pdf : nlpp.pdf

print : nlpp.pdf
        lpr nlpp.pdf

view : nlpp.pdf
        evince nlpp.pdf

◇
```
Fragment defined by ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `pdf` ?, ?, ?, `print` ?, ?, ?, ?, `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* ? ⟩ ≡
```
%.pdf : %.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

◇
```
Fragment defined by ?, ?, ?.
Fragment referenced in ?.
Uses: `pdf` ?, `PDFT_NAMES` ?, `PDF_FIG_NAMES` ?.

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* ? ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `nuweb` ?.

⟨ *parameters in Makefile* ? ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `nuweb` ?.

⟨ *expliciete make regels* ? ⟩ ≡
```
$(W2PDF) : nlpp.w $(NUWEB)
        $(NUWEB) nlpp.w
```
◇

Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `NUWEB` ?.

"../nuweb/bin/w2pdf" ?≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20150323 at 1045h: Generated by nuweb from a_nlpp.w
NUWEB=/Users/paul/bin/nuweb

LATEXCOMPILER=pdflatex
```
⟨ *filenames in nuweb compile script* ? ⟩
⟨ *compile nuweb* ? ⟩

◇

Uses: `NUWEB` ?, `nuweb` ?.

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* ? ⟩ ≡
```
NUWEB=/Users/paul/bin/nuweb
```

⟨ *run the processors until the aux file remains unchanged* ? ⟩
⟨ *remove the copy of the aux file* ? ⟩
◇

Fragment referenced in ?.
Uses: `NUWEB` ?, `nuweb` ?.

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

Fragment referenced in ?.
Defines: `auxfil` ?, ?, ?, `indexfil` ?, ?, `nufil` ?, ?, ?, `oldaux` ?, ?, ?, ?, `oldindexfil` ?, ?, `texfil` ?, ?, ?, `trunk` ?, ?, ?, ?.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file ?* ⟩ ≡

```
rm $oldaux
◇
```

Fragment referenced in ?, ?.
Uses: `oldaux` ?, ?.

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 ?* ⟩ ≡

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

Fragment referenced in ?.
Defines: `bibtex` ?, ?, `makeindex` ?, ?, `nuweb` ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?.
Uses: `nufil` ?, ?, `NUWEB` ?, `texfil` ?, ?, `trunk` ?, ?.

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 ?* ⟩ ≡

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

Fragment referenced in ?.
Uses: `auxfil` ?, ?, `indexfil` ?, `oldaux` ?, ?, `oldindexfil` ?.

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`.

To create a HTML doc, we do the following:

1. Create a directory `../nuweb/html` for the HTML document.
2. Put the nuweb source in it, together with style-files that are needed (see variable `HTMLSOURCE`).
3. Put the script `w2html` in it and make it executable.
4. Execute the script `w2html`.

Make a list of the entities that we mentioned above:

⟨ *parameters in Makefile* ? ⟩ ≡

```
htmldir=../nuweb/html
htmlsource=nlpp.w nlpp.bib html.sty artikel3.4ht w2html
htmlmaterial=$(foreach fil, $(htmlsource), $(htmldir)/$(fil))
htmltarget=$(htmldir)/nlpp.html
◇
```

Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `nuweb` ?.

Make the directory:

⟨ *expliciete make regels* ? ⟩ ≡

```
$(htmldir) :
        mkdir -p $(htmldir)

◇
```

Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

The rule to copy files in it:

⟨ *impliciete make regels* ? ⟩ ≡

```
$(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/

◇
```

Fragment defined by ?, ?, ?.
Fragment referenced in ?.

Do the work:

⟨ *expliciete make regels* ? ⟩ ≡

```
$(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w

◇
```

Fragment defined by ?, ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.

Invoke:

⟨ *make targets* ? ⟩ ≡

```
htm : $(htmldir) $(htmltarget)

◇
```

Fragment defined by ?, ?, ?, ?.
Fragment referenced in ?.

Create a script that performs the translation.

"w2html" ?≡
```
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
#  [filename]: Name of the nuweb source file.
# 20150323 at 1045h: Generated by nuweb from a_nlpp.w
echo "translate " $1 >w2html.log
NUWEB=/Users/paul/bin/nuweb

⟨ filenames in w2html ? ⟩

⟨ perform the task of w2html ? ⟩

◇
```
Uses: `NUWEB` ?, `nuweb` ?.

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* ? ⟩ ≡
```
⟨ run the html processors until the aux file remains unchanged ? ⟩
⟨ remove the copy of the aux file ? ⟩
◇
```
Fragment referenced in ?.

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* ? ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in ?.
Defines: `auxfil` ?, ?, ?, `nufil` ?, ?, ?, `oldaux` ?, ?, ?, ?, `texfil` ?, ?, ?, `trunk` ?, ?, ?, ?.
Uses: `indexfil` ?, `oldindexfil` ?.

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

◇
```
Fragment referenced in ?.
Uses: `auxfil` ?, ?, `oldaux` ?, ?.

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

⟨ *run the html processors* ? ⟩ ≡
```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in ?.
Uses: `bibtex` ?, `makeindex` ?, `nufil` ?, ?, `NUWEB` ?, `texfil` ?, ?, `trunk` ?, ?.

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* ? ⟩ ≡
```
tex '\def\filename{{nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in ?.
Uses: `bibtex` ?, `makeindex` ?, `trunk` ?, ?.

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* ? ⟩ ≡
```
MKDIR = mkdir -p


◇
```
Fragment defined by ?, ?, ?, ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `MKDIR` ?.

⟨ *make targets* ? ⟩ ≡
```
DIRS = ⟨ directories to create ?, ... ⟩

$(DIRS) :
        $(MKDIR) $@

◇
```
Fragment defined by ?, ?, ?, ?.
Fragment referenced in ?.
Defines: `DIRS` ?.
Uses: `MKDIR` ?.

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

⟨ *make targets* ? ⟩ ≡
```
sources : nlpp.w $(DIRS) $(NUWEB)
        $(NUWEB) nlpp.w
        ⟨ make scripts executable ? ⟩

◇
```
Fragment defined by ?, ?, ?, ?.
Fragment referenced in ?.
Uses: `DIRS` ?, `NUWEB` ?.

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

C.3 Variables