Bilingual NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

30th May 2016
14:24 h.

Abstract

This is a description and documentation of the installation of an instrument to annotate Dutch or English documents with NLP tags.

Contents

1 Introduction

This document describes the current set-up of a pipeline that annotates 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] project. It accepts and produces texts in the NAF (Newsreader Annotation Format) format.

Apart from describing the pipeline set-up, the document actually constructs the pipeline. The pipeline has been installed on a (Ubuntu) Linux computer.

The installation has been parameterised. 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.

The pipeline is bi-lingual. It is capable to annotate Dutch and English texts. It recognizes the language from the "lang" attribute of the `NAF` element of the document.

The aim is, to install the pipeline from open-source modules that can e.g. be obtained from Github. However, that aim is only partially fulfilled. Some of the modules still contain elements that are not open-source ot data that are not freely available. Because of lack of time, the current version of the installer installs the English pipeline from a frozen repository of the Newsreader Project.

1.1 List of the modules to be installed

Table 2 lists the modules in the pipeline. The column *source* indicates the origin of the module. The modules are obtained in one of the following ways:

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

The modules themselves use other utilities like dependency-taggers and POS taggers. These utilities are listed in table 1.

1.2 The things that are not open-source yet

The aim is, that the pipeline-system is completely open-sourced, so that anybody can install it from sources like Github. Howver, a lot of elements are not yet open-sourced, but need private kludges. The following is a list of not-yet open things.

---

1. http://wordpress.let.vupr.nl
2. http://www.newsreader-project.eu

Module	Version	Section	Source
KafNafParserPy	Feb 1, 2015	4.4.3	Github
Alpino	20706	5.5.4	RUG
Ticcutils	0.7	5.5.6	ILK
Timbl	6.4.6	5.5.6	ILK
Treetagger	3.2	5.5.5	Uni. München
Spotlight server	0.7	5.5.8	Spotlight

Table 1: *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.*

Module	Source	Section	Commit	Script	language
Tokenizer	https://github.com/ixa-ehu/ixa-pipe-tok.git	5.6.1	56f8...	tok	en/nl
Topic detection	https://github.com/ialdabe/ixa-pipe-topic.git	5.6.2	40be...	topic	en/nl
Morpho-syntactic parser	https://github.com/cltl/morphosyntactic_parser_nl.git	5.6.3	d5f0...	mor	nl
POS-tagger	snapshot	5.6.4	...	pos	en
Named-entity rec/class	https://github.com/ixa-ehu/ixa-pipe-nerc	5.6.17	ca02...	nerc	en/nl
Constituent parser	snapshot	5.6.5	...	constpars	en
Word-sense disamb. nl	https://github.com/cltl/svm_wsd.git	5.6.18	0300...	wsd	nl
Word-sense disamb. en	snapshot	5.6.9	...	ewsd	en
Named entity/DBP	snapshot	5.6.20	...	ned	en/nl
NED reranker	snapshot	5.6.6	...	nedrerscript	en
Wikify	snapshot	5.6.7	...	wikify	en
UKB	snapshot	5.6.8	...	ukb	en
Coreference-base	snapshot	5.6.16	...	coreference-base	en
Heideltime	https://github.com/ixa-ehu/ixa-pipe-time.git	5.6.22	da46...	heideltime	nl
Onto-tagger	https://github.com/cltl/OntoTagger.git	5.6.21	9ea0...	onto	nl
Semantic Role labeling nl	https://github.com/newsreader/vua-srl-nl.git	5.6.23	675d...	srl	nl
Semantic Role labeling en	snapshot	5.6.10	...	eSRL	en
Nominal Event ann.	https://github.com/cltl/OntoTagger.git	5.6.21	9ea0...	nomevent	nl
SRL dutch nominals	https://github.com/newsreader/vua-srl-dutch-nominal-events	5.6.11	6115...	srl-dutch-nominals	nl
Framenet-SRL	https://github.com/cltl/OntoTagger.git	5.6.21	9ea0...	framesrl	nl
FBK-time	snapshot	5.6.12	...	FBK-time	en
FBK-temprel	snapshot	5.6.13	...	FBK-temprel	en
FBK-causalrel	snapshot	5.6.14	...	FBK-causalrel	en
Opinion-miner	https://github.com/rubenIzquierdo/opinion_miner_deluxePP	5.6.27	5f46...	opinimin	en/nl
Event-coref	snapshot	5.6.25	...	evcoref	en/nl
Factuality tagger	snapshot	5.6.15	...	factuality	en

Table 2: *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.*

1.3 Multi-linguality

Thi version of the pipeline is multi-lingual, i.e. it can annotate Dtutsch as well as English documents. It finds the language of the document in the `language` attribute of the `NAF` element. Actually, the current version is bi-lingual, because it is only able to process Dutch or English documents.

1.4 File-structure of the pipeline

The files that make up the pipeline are organised in set of directories as shown in figure 1. The

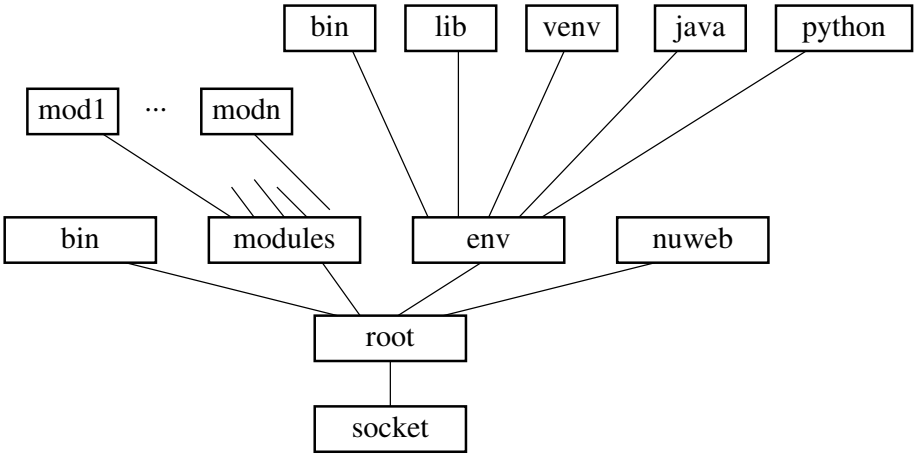

Figure 1: *Directory-structure of the pipeline (see text).*

directories have the follosing functions.

socket: The directory in the host where the pipeline is to be implemented.
root: The root of the pipeline directory-structure.
nuweb: This directory contains this document and everything to create the pipeline from the open sources of the modules.
modules: Contains subdirectories with the NLP modules that can be applied in the pipeline.
bin: Contains for each of the applicable 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.
env: The programming environment. It contains a.o. the Java development kit, Python, the Python virtual environment (`venv`), libraries and binaries.

⟨ *directories to create* 6a ⟩ ≡
```
../modules ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

⟨ *directories to create* 6b ⟩ ≡
```
../bin ../env/bin ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

⟨ *directories to create* 7a ⟩ ≡
```
../env/lib ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

⟨ *directories to create* 7b ⟩ ≡
```
../env/etc ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

The following macro defines variable `piperoot` and makes it to point to the root directory in figure 1. Next it defines variables that point to other directories in the figure. The value-setting of `piperoot` can be overruled by defining the variable before running any of the script. In this way the directory tree can be moved to another location, even to another computer, after successful installation.

⟨ *set variables that point to the directory-structure* 7c ⟩ ≡
```
if
  [ "$piperoot" == "" ]
then
  export piperoot=/home/huygen/projecten/nlpp
fi
export pipesocket=${piperoot%%/nlpp}
export nuwebdir=$piperoot/nuweb
export envdir=$piperoot/env
export envbindir=$envdir/bin
export envlibdir=$envdir/lib
export modulesdir=$piperoot/modules
export pipebin=$piperoot/bin
export javadir=$envdir/java
export jarsdir=$javadir/jars
◇
```
Fragment defined by 7cd, 9f, 13b.
Fragment referenced in 7e, 20a, 82c.
Uses: `nuweb` 77b.

Add the environment `bin` directory to `PATH`:

⟨ *set variables that point to the directory-structure* 7d ⟩ ≡
```
export PATH=$envbindir:$PATH
◇
```
Fragment defined by 7cd, 9f, 13b.
Fragment referenced in 7e, 20a, 82c.
Defines: `PATH` 12f, 13bf, 17c, 51d.

Put the macro to set variables in a script that can later be sourced by the scripts of the pipeline modules.

`"../env/bin/progenv"` 7e≡
```
#!/bin/bash
⟨ set variables that point to the directory-structure 7c, ... ⟩
export progenvset=0
◇
```
File defined by 7e, 11f.

2 How to obtain modules and other material

As illustrated in tables 2 and 1, most of the modules are obtained as source-code from Github, some of the modules or parts of some modules are downloaded from a snapshot, and some of the utilities are obtained in binary form from the supplier.

This section builds standardised methods to obtain modules and utilities from Github or from the snapshot.

2.1 Location-dependency

The basic way of installation is, to clone this repository from Github on the intended location in the file-system of the target computer and then run the install-scripts. However, it may be advantageous to be able to transplant a complete installation to another location in another computer. This could be done by making all path-descriptions in all scripts relative to anchorpoints within the installation, while it may be hard to find such anchorpoints in advance. Therefore, we take another approach in which we supply a script that repairs paths-descriptions after the transplantation (section A.9).

2.2 Reversible update

This script might be used to update an existing installation. To minimize the risk that the "update" acually ruins an existing installation, move existing modules away before installing the latest version. When the new modules has been installed succesfully, the moved module will be removed. The following macro's help to achieve this:

⟨ *move module* 8a ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
```
◇

Fragment referenced in 9a, 68c.

⟨ *remove old module* 8b ⟩ ≡
```
rm -rf old.@1
```
◇

Fragment referenced in 9a, 68c.

⟨ *re-instate old module* 8c ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
```
⟨ *logmess* (8d `$MESS`) 68b ⟩

◇

Fragment referenced in 9a, 68c.

2.3 Installation from Github

The following macro can be used to install a module from Github. Before issuing this macro, 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* 9a ⟩ ≡
```
cd $modulesdir
⟨ move module (9b $DIRN ) 8a ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (9c Cannot install current $MODNAM version ) 68b ⟩
  ⟨ re-instate old module (9d $DIRN ) 8c ⟩
else
  ⟨ remove old module (9e $DIRN ) 8b ⟩
  cd $modulesdir/$DIRN
  git checkout $GITC
fi

◇
```
Fragment referenced in 41d, 46g, 54a, 55d, 58b, 59e, 62c, 63a.

2.4 Installation from the snapshot

The sources for the non-open parts of the pipeline are collected in directory `t_nlpp_resources`. They can be accessed via SSH from url `m4_snapshotURL`. Before installing the pipeline download the snapshot on top of directory `snapshotsocket`.

⟨ *set variables that point to the directory-structure* 9f ⟩ ≡
```
if
  [ ! $snapshotsocket ]
then
  export snapshotsocket=/home/huygen/projecten
fi
◇
```
Fragment defined by 7cd, 9f, 13b.
Fragment referenced in 7e, 20a, 82c.

The snapshot can be accessed over `scp` on URL `newsreader@kyoto.let.vu.nl`. Access is protected by a public/private key system. So, a private key is needed and this program expects to to find the key as `$pipesocket/nrkey`. The key can be obtained from the author. Let us check whether we indeed do have the key:

⟨ *check this first* 9g ⟩ ≡
```
if
  [ ! -e /home/huygen/projecten/nrkey ]
then
  echo "No key to connect to snapshot!"
  exit 1
fi
◇
```
Fragment defined by 9g, 24e.
Fragment referenced in 20a.

Update the local snapshot repository.

⟨ *get the snapshot* 10a ⟩ ≡

```
cd $snapshotsocket
rsync -e "ssh -i /home/huygen/projecten/nrkey" -
rLt newsreader@kyoto.let.vu.nl:t_nlpp_resources .
◇
```

Fragment referenced in 20a.

3 Shared libraries

When we do not want to rely on what the host can present to us, we need to make our own shared libraries. For the present, we will generate the shared libraries `libxslt` and `libxml2`. We do the following:

1. install autoconf, needed to compile the libs.
2. install libxslt
3. install libxml2

3.1 Autoconf

Gnu autoconf is a system to help configure the Makefiles for a software package. Softwarepackages that use this, supply a file `configure`, `configure.in` or `configure.ac`. To compile and install a package from source we can then perform 1) `./configure --prefix=<environment>`; 2) `make`; 3) `make install`.

Install autoconf:

⟨ *install shared libs* 10b ⟩ ≡

```
autoconfdir=‘mktemp -d -t autoconf.XXXXXX‘
cd $autoconfdir
wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz
tar -xzf autoconf-2.69.tar.gz
cd autoconf-2.69
./autogen.sh --prefix=$envdir
make
make install
cd $piperoot
rm -rf $autoconfdir
◇
```

Fragment defined by 10b, 11b.
Fragment referenced in 20a.
Uses: `install` 81d.

3.2 libxml2 and libxslt

Compilation and installation of `libxml2` and `libxslt` goes similar, according to the following template:

⟨ *install libxml2 or libxslt* 11a ⟩ ≡

```
shtmpdir=‘mktemp -d -t shl.XXXXXX‘
cd $shtmpdir
git clone @1
packagedir=‘ls -1‘
cd $packagedir
./autogen.sh --prefix=$envdir
make
make install
cd $piperoot
rm -rf $shtmpdir
```

◇

Fragment referenced in 11b.
Uses: `install` 81d.

⟨ *install shared libs* 11b ⟩ ≡

⟨ *install libxml2 or libxslt* (11c `git://git.gnome.org/libxml2`) 11a ⟩
⟨ *install libxml2 or libxslt* (11d `git://git.gnome.org/libxslt`) 11a ⟩
◇

Fragment defined by 10b, 11b.
Fragment referenced in 20a.

4 Java, Python en Perl

To be independent from the software environment of the host computer and to perform reproducible processing, the pipeline features its own Java, Perl and Python environments. 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 macro generates a script that specifies the programming environment. Initially it is empty, because we have to create the programming environment first.

⟨ *create javapython script* 11e ⟩ ≡

```
echo '#!/bin/bash' > /home/huygen/projecten/nlpp/env/bin/javapython
```

◇

Fragment referenced in 20a.

Cause the module scripts to read the javapython script.

`"../env/bin/progenv"` 11f≡

```
source $envbindir/javapython
```

◇

File defined by 7e, 11f.

4.1 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 `envdir`.

⟨ *directories to create* 12a ⟩ ≡
```
../env/java ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

⟨ *set up java* 12b ⟩ ≡
```
⟨ begin conditional install (12c java_installed ) 18c ⟩
  cd $envdir/java
  tar -xzf $snapshotsocket/t_nlpp_resources/server-jre-7u72-linux-x64.tar.gz
⟨ end conditional install (12d java_installed ) 19b ⟩
◇
```
Fragment defined by 12bf.
Fragment referenced in 20a.

Remove the java-ball when cleaning up:

⟨ *clean up* 12e ⟩ ≡
```
rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz
◇
```
Fragment defined by 12e, 13c, 28d, 72a.
Fragment referenced in 71a.

Set variables for Java.

⟨ *set up java* 12f ⟩ ≡
```
echo 'export JAVA_HOME=$envdir/java/jdk1.7.0_72' >> /home/huygen/projecten/nlpp/env/bin/javapython
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /home/huygen/projecten/nlpp/env/bin/javapython
export JAVA_HOME=$envdir/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 12bf.
Fragment referenced in 20a.
Uses: PATH 7d.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 12g ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

4.2 Maven

Some Java-based modules can best be compiled with Maven.

⟨ *directories to create* 12h ⟩ ≡
```
../env/apache-maven-3.0.5 ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

⟨ *install maven* 13a ⟩ ≡
```
cd $envdir
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 referenced in 20a.

⟨ *set variables that point to the directory-structure* 13b ⟩ ≡
```
export MAVEN_HOME=$envdir/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
```
◇

Fragment defined by 7cd, 9f, 13b.
Fragment referenced in 7e, 20a, 82c.
Uses: PATH 7d.

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

⟨ *clean up* 13c ⟩ ≡
```
rm -rf ../env/apache-maven-3.0.5
```
⟨ *remove installed-variable* (13d `maven_installed`) 19c ⟩
◇

Fragment defined by 12e, 13c, 28d, 72a.
Fragment referenced in 71a.

4.3 Java 1.6

Java 1.7 is able to run nearly all the modules of the pipeline that are based on Java. However, there is one exception, i.e. the `ims-wsd` module, that needs Java version 1.6. So, we have to install that version of Java as well.

⟨ *install Java 1.6* 13e ⟩ ≡
```
cd $envdir/java
$snapshotsocket/t_nlpp_resources/jre-6u45-linux-x64.bin
```
◇

Fragment referenced in 20a.

Insert the following macro in scripts that need to run Java 1.6.

⟨ *set up Java 1.6* 13f ⟩ ≡
```
export JAVA_HOME=$envdir/java/jre1.6.0_45
export PATH=$JAVA_HOME/bin:$PATH
```
◇

Fragment referenced in 45b.
Uses: PATH 7d.

4.4 Python

Set up the environment for Python (version 2.7). I could not find an easy way to set up Python from scratch. Therefore we wil use Python 2.7 if is has been installed on the host. Otherwise,

we will use a binary distribution obtained from ActiveState. A tarball of ActivePython can be obtained from the snapshot.

In order to be independent of the software on the host, we generate a virtual Python environment. In the virtual environment we will install KafNafParserPy and other Python packages that are needed.

⟨ *set up python* 14a ⟩ ≡
⟨ *check/install the correct version of python* 14b ⟩
⟨ *create a virtual environment for Python* 15a ⟩
⟨ *activate the python environment* 15c, ... ⟩
⟨ *update pip* 16b ⟩
⟨ *install python packages* 17a, ... ⟩
⟨ *install kafnafparserpy* 16d ⟩
◇
Fragment referenced in 20a.

⟨ *check/install the correct version of python* 14b ⟩ ≡
```
pythonok=‘python --
version 2>&1 | gawk ’{if(match($2, "2.7")) print "yes" ; else print "no" }’‘
if
  [ "$pythonok" == "no" ]
then
  ⟨ install ActivePython 14c ⟩
fi
```
◇
Fragment referenced in 14a.
Defines: `pythonok` Never used.
Uses: `print` 75a.

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* 14c ⟩ ≡
```
pytinsdir=‘mktemp -d -t activepyt.XXXXXX‘
cd $pytinsdir
tar -xzf $snapshotsocket/t_nlpp_resources/ActivePython-2.7.8.10-linux-x86_64.tar.gz
acdir=‘ls -1‘
cd $acdir
./install.sh -I $envdir
cd $piperoot
rm -rf $pytinsdir
pip install -U virtualenv setuptools
```
◇
Fragment referenced in 14b.
Uses: `install` 81d, `virtualenv` 15b.

4.4.1 Virtual environment

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

⟨ *create a virtual environment for Python* 15a ⟩ ≡
⟨ *test whether virtualenv is present on the host* 15b ⟩
```
cd $envdir
virtualenv venv
```
◇
Fragment referenced in 14a.
Uses: virtualenv 15b.

⟨ *test whether virtualenv is present on the host* 15b ⟩ ≡
```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
```
◇
Fragment referenced in 15a.
Defines: virtualenv 14c, 15a.
Uses: install 81d.

Activate the virtual environment immediately in the installation-script, and add the activation-instruction to the initialisation-script.

⟨ *activate the python environment* 15c ⟩ ≡
```
source $envdir/venv/bin/activate
echo 'source $en-
vdir/venv/bin/activate' >> /home/huygen/projecten/nlpp/env/bin/javapython
```
◇
Fragment defined by 15ce, 16a.
Fragment referenced in 14a, 20a.
Defines: activate 16c.

Subdirectory `$envdir/python` will contain general Python packages like KafnafParserPy.

⟨ *directories to create* 15d ⟩ ≡
```
../env/python
```
◇
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.

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

⟨ *activate the python environment* 15e ⟩ ≡
```
echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /home/huygen/projecten/nlpp/env/bin/javapython
export PYTHONPATH=$envdir/python:$PYTHONPATH
```
◇
Fragment defined by 15ce, 16a.
Fragment referenced in 14a, 20a.
Defines: PYTHONPATH Never used.

We will use home-brewed shared libraries in Python, e.g. libxml2 and libxslt:

⟨ *activate the python environment* 16a ⟩ ≡
```
echo ex-
port 'LD_LIBRARY_PATH=$envlibdir:$LD_LIBRARY_PATH' >> /home/huygen/projecten/nlpp/env/bin/javapython
export LD_LIBRARY_PATH=$envdir/python:$LD_LIBRARY_PATH
◇
```
Fragment defined by 15ce, 16a.
Fragment referenced in 14a, 20a.
Defines: `LD_LIBRARY_PATH` 44d, 51d.

Update pip in the virtual environment, because otherwise it keeps complaining about outdated versions

⟨ *update pip* 16b ⟩ ≡
```
pip install --upgrade pip
◇
```
Fragment referenced in 14a.
Uses: `install` 81d.

4.4.2 Transplant the virtual environment

It turns out that the script "activate" to engage the virtual environment contains an absolute path, in the definition of `VIRTUAL_ENV`

⟨ *set paths after transplantation* 16c ⟩ ≡
```
transdir=‘mktemp -d -t trans.XXXXXX‘
cd $transdir
cat <<EOF >redef.awk
#!/usr/bin/gawk -f
BEGIN { envd="$envdir/venv"}

/^VIRTUAL_ENV=/ { print "VIRTUAL_ENV=\"" envd "\""
                  next
                }
{print}
EOF

mv $envdir/venv/bin/activate .
gawk -f redef.awk ./activate > $envdir/venv/bin/activate
cd $projroot
rm -rf $transdir
◇
```
Fragment referenced in 82c.
Uses: `activate` 15c, `print` 75a.

4.4.3 KafNafParserPy

A cornerstone Pythonmodule for the pipeline is KafNafParserPy. Currently it is extremely easy installed:

⟨ *install kafnafparserpy* 16d ⟩ ≡
```
pip install KafNafParserPy
◇
```
Fragment referenced in 14a.
Uses: `install` 81d.

4.4.4 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph
pynaf:
requests: for networkx
networkx: for corefbase.

⟨ *install python packages* 17a ⟩ ≡

```
pip install lxml
pip install pyyaml
pip install --upgrade git+https://github.com/ixa-ehu/pynaf.git
pip install --upgrade requests
pip install --upgrade networkx
◇
```

Fragment defined by 17a, 61a.
Fragment referenced in 14a.
Defines: `lxml` Never used, `networkx` Never used, `pyyaml` Never used.
Uses: `install` 81d.

4.5 Perl

Install Perl locally, to be certain that Perl is available and to enable to install packages that we need (in any case: `XML::LibXML`).

⟨ *install perl* 17b ⟩ ≡

```
tempdir=‘mktemp -d -t perl.XXXXXX‘
cd $tempdir
wget http://www.cpan.org/src/5.0/perl-5.22.1.tar.gz
tar -xzf perl-5.22.1.tar.gz
cd perl-5.22.1
./Configure -des -Dprefix=$envdir/perl
make
make test
make install
cd $progroot
rm -rf $tempdir

◇
```

Fragment defined by 17bc, 18a.
Fragment referenced in 20a.
Uses: `install` 81d.

Make sure that modules use the correct Perl

⟨ *install perl* 17c ⟩ ≡

```
echo 'export PERL_HOME=$envdir/perl' >> /home/huygen/projecten/nlpp/env/bin/javapython
echo 'export PATH=$PERL_HOME/bin:$PATH' >> /home/huygen/projecten/nlpp/env/bin/javapython
export PERL_HOME=$envdir/perl
export PATH=$PERL_HOME/bin:$PATH
◇
```

Fragment defined by 17bc, 18a.
Fragment referenced in 20a.
Uses: `PATH` 7d.

Install what is called `XML::XMLLib` in the Perl world.

It should be done with the following statement:

```
perl -MCPAN -e 'install XML::LibXML'
```

but that doesn't seem to work in all cases. It worked during an installation in ArchLinux, but not in an installation in Ubuntu a few weeks later.

Therefore, get the lib from the snapshot.

⟨ *install perl* 18a ⟩ ≡

```
cd $envdir/perl/lib
tar -xzf $snapshotsocket/t_nlpp_resources/20160520_nlpp_perllib.tgz
◇
```

Fragment defined by 17bc, 18a.
Fragment referenced in 20a.

5 Installation of the modules

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

5.1 Conditional installation of the modules

Next section generates a script that installs everything.

Installation is very time-intensive. To prevent that everything is re-installed every time that the module-installer is run, there is a list of variables, the *modulelist*, that are set when a module has been installed. To re-install that module, remove the variable from the list and then re-run the installer. It maintains a list of the modules and utilitie that is has installed and installs only modules and utilities that are not on the list. So in order to re-install a module that has already been installed, remove it from the list and then re-run the module-installer.

The modulelist is in fact a script named `/home/huygen/projecten/nlpp/installed_modules` that sets Bash variables. It ought to be sourced if it is present.

Initially the list is not present. When a module or a utility has been installed, an instruction to set a variable is written in or appended to the list.

⟨ *read the list of installed modules* 18b ⟩ ≡

```
if
  [ -e /home/huygen/projecten/nlpp/installed_modules ]
then
  source /home/huygen/projecten/nlpp/installed_modules
fi
◇
```

Fragment referenced in 20a.

⟨ *begin conditional install* 18c ⟩ ≡

```
if
  [ ! $@1 ]
then
◇
```

Fragment referenced in 12b, 20a, 21aj, 22aj, 23ahq, 24a.

⟨ *else conditional install* 19a ⟩ ≡
```
else
```
◇
Fragment never referenced.

⟨ *end conditional install* 19b ⟩ ≡
```
  echo "export @1=0" >> /home/huygen/projecten/nlpp/installed_modules
fi
```
◇
Fragment referenced in 12b, 20a, 21aj, 22aj, 23ahq, 24a.

Remove a variable from the list of installed modules, e.g. after a clean-up.

⟨ *remove installed-variable* 19c ⟩ ≡
```
cd $piperoot
mv /home/huygen/projecten/nlpp/installed_modules old.modulelist
cat old.modulelist | gawk '/@1/ {next}; {print}' >/home/huygen/projecten/nlpp/installed_modules
```
◇
Fragment referenced in 13c.
Uses: `print` 75a.

5.2 The installation script

The installation is performed by script `install-modules`.

The first part of the script installs the utilities:

"../bin/install-modules" 20a≡
```
#!/bin/bash
echo Set up environment
⟨ set variables that point to the directory-structure 7c, ... ⟩
⟨ read the list of installed modules 18b ⟩
⟨ check this first 9g, ... ⟩
⟨ begin conditional install (20b repo_installed ) 18c ⟩
  ⟨ get the snapshot 10a ⟩
⟨ end conditional install (20c repo_installed ) 19b ⟩
⟨ variables of install-modules 68a ⟩
⟨ begin conditional install (20d shared_libs ) 18c ⟩
  ⟨ install shared libs 10b, ... ⟩
⟨ end conditional install (20e shared_libs ) 19b ⟩
⟨ create javapython script 11e ⟩
echo ... Java
⟨ set up java 12b, ... ⟩
⟨ begin conditional install (20f maven_installed ) 18c ⟩
  ⟨ install maven 13a ⟩
⟨ end conditional install (20g maven_installed ) 19b ⟩
⟨ begin conditional install (20h java16_installed ) 18c ⟩
  ⟨ install Java 1.6 13e ⟩
⟨ end conditional install (20i java16_installed ) 19b ⟩

echo ... Python
if
  [ $python_installed ]
then
  ⟨ activate the python environment 15c, ... ⟩
fi
⟨ begin conditional install (20j python_installed ) 18c ⟩
  ⟨ set up python 14a ⟩
⟨ end conditional install (20k python_installed ) 19b ⟩
⟨ begin conditional install (20l perl_installed ) 18c ⟩
  ⟨ install perl 17b, ... ⟩
⟨ end conditional install (20m perl_installed ) 19b ⟩

⟨ begin conditional install (20n sematree_installed ) 18c ⟩
  ⟨ install sematree 26a ⟩
⟨ end conditional install (20o sematree_installed ) 19b ⟩
echo ... Alpino
⟨ begin conditional install (20p alpino_installed ) 18c ⟩
  ⟨ install Alpino 28b ⟩
⟨ end conditional install (20q alpino_installed ) 19b ⟩
echo ... Spotlight
⟨ begin conditional install (20r spotlight_installed ) 18c ⟩
  ⟨ install the Spotlight server 32a, ... ⟩
⟨ end conditional install (20s spotlight_installed ) 19b ⟩
echo ... Treetagger
⟨ begin conditional install (20t treetagger_installed ) 18c ⟩
  ⟨ install the treetagger utility 29a, ... ⟩
⟨ end conditional install (20u treetagger_installed ) 19b ⟩
echo ... Ticcutils and Timbl
⟨ begin conditional install (20v ticctimbl_installed ) 18c ⟩
  ⟨ install the ticcutils utility 30d ⟩
  ⟨ install the timbl utility 31a ⟩
⟨ end conditional install (20w ticctimbl_installed ) 19b ⟩
echo ... Boost
⟨ begin conditional install (20x boost_installed ) 18c ⟩
  ⟨ install boost 31d ⟩
⟨ end conditional install (20y boost_installed ) 19b ⟩
echo ... VUA-pylib, SVMlight, CRFsuite
⟨ begin conditional install (20z miscutils_installed ) 18c ⟩
  ⟨ install VUA-pylib 38a ⟩
  ⟨ install SVMLight 38b ⟩
  ⟨ install CRFsuite 39a ⟩
⟨ end conditional install (20 miscutils_installed ) 19b ⟩◇
```

File defined by 20a, 21a, 22a, 23abc, 24a.

Next, install the modules:

```
"../bin/install-modules" 21a≡
    echo Install modules
    ⟨ begin conditional install (21b tokenizer_installed ) 18c ⟩
      echo ... Tokenizer
      ⟨ install the tokenizer 39b ⟩
    ⟨ end conditional install (21c tokenizer_installed ) 19b ⟩
    ⟨ begin conditional install (21d topic_installed ) 18c ⟩
      echo ... Topic detector
      ⟨ install the topic analyser 40a ⟩
    ⟨ end conditional install (21e topic_installed ) 19b ⟩
    ⟨ begin conditional install (21f morpar_installed ) 18c ⟩
      echo ... Morphosyntactic parser
      ⟨ install the morphosyntactic parser 41d ⟩
    ⟨ end conditional install (21g morpar_installed ) 19b ⟩
    ⟨ begin conditional install (21h pos_installed ) 18c ⟩
      echo "... Pos tagger (for english docs)"
      ⟨ install the pos tagger 42b ⟩
    ⟨ end conditional install (21i pos_installed ) 19b ⟩
    ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
"../bin/install-modules" 21j≡
    ⟨ begin conditional install (21k constparse_installed ) 18c ⟩
      echo "... Constituent parser (for english docs)"
      ⟨ install the constituents parser 43a ⟩
    ⟨ end conditional install (21l constparse_installed ) 19b ⟩
    ⟨ begin conditional install (21m nerc_installed ) 18c ⟩
      echo ... NERC
      ⟨ install the NERC module 52d ⟩
    ⟨ end conditional install (21n nerc_installed ) 19b ⟩
    ⟨ begin conditional install (21o ned_installed ) 18c ⟩
      echo ... NED
      ⟨ install the NED module 55d ⟩
    ⟨ end conditional install (21p ned_installed ) 19b ⟩
    ⟨ begin conditional install (21q nedrer_installed ) 18c ⟩
      echo ...NED reranker
      ⟨ install the NED-reranker module 43d ⟩
    ⟨ end conditional install (21r nedrer_installed ) 19b ⟩
    ⟨ begin conditional install (21s wikify_installed ) 18c ⟩
      echo ...WIKIfy module
      ⟨ install the wikify module 43g ⟩
    ⟨ end conditional install (21t wikify_installed ) 19b ⟩
    ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

"../bin/install-modules" 22a≡
⟨ *begin conditional install* (22b `UKB_installed`) 18c ⟩
```
  echo ... UKB module
 cd $modulesdir
 tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-ukb.v30.tgz
```
⟨ *end conditional install* (22c `UKB_installed`) 19b ⟩
⟨ *begin conditional install* (22d `ims_wsd_installed`) 18c ⟩
```
  echo ...ims-wsd module
```
⟨ *install the ims-wsd module* 45a ⟩
⟨ *end conditional install* (22e `ims_wsd_installed`) 19b ⟩
⟨ *begin conditional install* (22f `srl_server_installed`) 18c ⟩
```
  echo ...srl-server module
```
⟨ *install the srl-server module* 45d ⟩
⟨ *end conditional install* (22g `srl_server_installed`) 19b ⟩
⟨ *begin conditional install* (22h `srl_dutch_nominals_installed`) 18c ⟩
```
  echo ...srl-dutch-nominal module
```
⟨ *install the srl-dutch-nominals module* 46g ⟩
⟨ *end conditional install* (22i `srl_dutch_nominals_installed`) 19b ⟩
◇

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

"../bin/install-modules" 22j≡
⟨ *begin conditional install* (22k `FBK_time_installed`) 18c ⟩
```
  echo ... FBK-time module
```
⟨ *install the FBK-time module* 47c ⟩
⟨ *end conditional install* (22l `FBK_time_installed`) 19b ⟩
⟨ *begin conditional install* (22m `FBK_temprel_installed`) 18c ⟩
```
  echo ... FBK-temprel module
```
⟨ *install the FBK-temprel module* 49b ⟩
⟨ *end conditional install* (22n `FBK_temprel_installed`) 19b ⟩
⟨ *begin conditional install* (22o `FBK_causalrel_installed`) 18c ⟩
```
  echo ... FBK-causalrel module
```
⟨ *install the FBK-causalrel module* 50c ⟩
⟨ *end conditional install* (22p `FBK_causalrel_installed`) 19b ⟩
⟨ *begin conditional install* (22q `factuality_installed`) 18c ⟩
```
  echo ... factuality module
```
⟨ *install the factuality module* 51c ⟩
⟨ *end conditional install* (22r `factuality_installed`) 19b ⟩
◇

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
"../bin/install-modules" 23a≡
    ⟨ begin conditional install (23b corefb_installed ) 18c ⟩
      echo ... Coreference base
      ⟨ install coreference-base 52a ⟩
    ⟨ end conditional install (23c corefb_installed ) 19b ⟩
    ⟨ begin conditional install (23d wsd_installed ) 18c ⟩
      echo ... WSD
      ⟨ install the WSD module 54a ⟩
    ⟨ end conditional install (23e wsd_installed ) 19b ⟩
    ⟨ begin conditional install (23f ontojar_installed ) 18c ⟩
      echo ... Ontotagger
      ⟨ install the ontotagger repository 57a ⟩
    ⟨ end conditional install (23g ontojar_installed ) 19b ⟩
    ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
"../bin/install-modules" 23h≡
    ⟨ begin conditional install (23i heidel_installed ) 18c ⟩
      echo ... Heideltime
      ⟨ install the heideltime module 58a ⟩
    ⟨ end conditional install (23j heidel_installed ) 19b ⟩
    ⟨ begin conditional install (23k SRL_installed ) 18c ⟩
       echo ... SRL
       ⟨ install the srl module 59e ⟩
    ⟨ end conditional install (23l SRL_installed ) 19b ⟩
    ⟨ begin conditional install (23m eventcoref_installed ) 18c ⟩
       echo ... Event-coreference
       ⟨ install the event-coreference module 61e ⟩
    ⟨ end conditional install (23n eventcoref_installed ) 19b ⟩
    ⟨ begin conditional install (23o lu2synset_installed ) 18c ⟩
       echo ... lu2synset
       ⟨ install the lu2synset converter 55a ⟩
    ⟨ end conditional install (23p lu2synset_installed ) 19b ⟩
    ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
"../bin/install-modules" 23q≡
    ⟨ begin conditional install (23r dbpner_installed ) 18c ⟩
      echo ... dbpedia-ner
      ⟨ install the dbpedia-ner module 62c ⟩
    ⟨ end conditional install (23s dbpner_installed ) 19b ⟩
    ⟨ begin conditional install (23t post_SRL_installed ) 18c ⟩
       echo ... post-SRL
       ⟨ install the post-SRL module 61b ⟩
    ⟨ end conditional install (23u post_SRL_installed ) 19b ⟩
    ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
"../bin/install-modules" 24a≡
        ⟨ begin conditional install (24b opimin_installed ) 18c ⟩
           echo ... opinion-miner
           ⟨ install the opinion-miner 63a, ... ⟩
        ⟨ end conditional install (24c opimin_installed ) 19b ⟩

        echo Final
        ◇
```

File defined by 20a, 21aj, 22aj, 23ahq, 24a.

```
⟨ make scripts executable 24d ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```

Fragment defined by 24d, 34g, 81b.
Fragment referenced in 81c.
Uses: install 81d.

5.3 Check availability of resources

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

```
⟨ check this first 24e ⟩ ≡
        ⟨ check whether program is present (24f git ) 24j ⟩
        ⟨ check whether program is present (24g tar ) 24j ⟩
        ⟨ check whether program is present (24h unzip ) 24j ⟩
        ⟨ check whether program is present (24i tcsh ) 24j ⟩
        ⟨ check whether mercurial is present 25a ⟩
        ◇
```

Fragment defined by 9g, 24e.
Fragment referenced in 20a.

```
⟨ check whether program is present 24j ⟩ ≡
        which @1
        if
          [ $? -ne 0 ]
        then
          echo Please install @1.
          exit 1
        fi
        ◇
```

Fragment referenced in 24e.
Uses: install 81d.

⟨ *check whether mercurial is present* 25a ⟩ ≡

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

Fragment referenced in 24e.
Defines: `hg` Never used.
Uses: `install` 81d.

5.4 Parameters in module-scripts

Some modules need parameters. All modules need a language specification. The language can be passed as exported variable `naflang`, but it can also be passed as argument `-l`. Furthermore, some modules need contact with a Spotlight server. With the arguments `-h` and `-b` the host and port of a running Spotlight-server can be passed.

The code to obtain command-line arguments in Bash has been obtained from Stackoverflow. The following fragment reads the arguments `-l language`, `-h spotlighthost` and `-p spotlightport`:

⟨ *get commandline-arguments* 25b ⟩ ≡

```
while [[ $# > 1 ]]
do
  key="$1"

  case $key in
    -l|--language)
      naflang="$2"
      shift # past argument
    ;;
    -h|--spothost)
      spotlighthost="$2"
      shift # past argument
    ;;
    -p|--spotport)
    spotlightport="$2"
    shift # past argument
    ;;
    *)
            # unknown option
    ;;
  esac
  shift # past argument or value
done
◇
```

Fragment referenced in 33b.
Uses: `naflang` 66a.

5.5 Install utilities and resources

5.5.1 Process synchronisation

We will see that we sometimes have to install server-applications. However, it is possible that multiple processes are running pipeline modules in parallel, and then it may occur that two instances of a module try to install the same server-application. Therefore, we must make sure that only one application at a time is able to start the server.

The program `sematree`, found at http://www.pixelbeat.org/scripts/sematree/ enables to do this. When invoked with argument "acquire", the name of a "lockfile" and a time to wait (-1 means "wait an indefinite time"), it checks whether the lockfile exists. If that is the case, it either waits or fails. When the lockfile is not (or no longer) present, sematree creates the lockfile.

When installing sematree, set the default directory for lock-files. We set this as a subdirectory of the `env` tree. However, in some cases, notably when running in a node in Lisa, we need a directory on the filesystem of the node itself.

⟨ *install sematree* 26a ⟩ ≡
```
cat $snapshotsocket/t_nlpp_resources/sematree | \
  sed "s|/var/run|/home/huygen/projecten/nlpp/env/etc/sematree|g" \
> $envbindir/sematree
chmod 775 $envbindir/sematree
◇
```
Fragment referenced in 20a.

5.5.2 Prefix of scripts that run modules

Each module will be run by a Bash script located in subdirectory `bin`. The start of these scrips will have similar content. Insert the following macro to include this similar content, with the name of the module-directory as argument:

⟨ *start of module-script* 26b ⟩ ≡
```
#!/bin/bash
⟨ get the path to the module-script 26c ⟩
source /home/huygen/projecten/nlpp/env/bin/progenv
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
ROOT=$piperoot
MODDIR=$modulesdir/@1
⟨ run in subshell when naflang is not known 27b ⟩
⟨ run only if language is English or Dutch 28a ⟩
◇
```
Fragment referenced in 39c, 41be, 42c, 43be, 44ad, 45be, 46ac, 47a, 48a, 50a, 51ad, 52b, 53ce, 54d, 55b, 56b, 57bdf, 59c, 60a, 61c, 62ad, 63d.

Set variable `scriptpath` to the full path of the script that is running, order to be able to re-run it.

⟨ *get the path to the module-script* 26c ⟩ ≡
```
scriptdir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
scriptname=${0##*/}
scriptpath=$scriptdir/$scriptname
◇
```
Fragment referenced in 26b.
Defines: `scriptpath` 27b.

5.5.3 Language detection

The following script `../env/bin/langdetect.py` discerns the language of a NAF document. If it cannot find that attribute it prints `unknown`. The macro `set the language variable` uses this script to set variable `naflang`. All pipeline modules expect that this veriable has been set.

"../env/bin/langdetect.py" 27a≡
```
#!/usr/bin/env python
# langdetect -- Detect the language of a NAF document.
#
import xml.etree.ElementTree as ET
import sys
import re
xmldoc = sys.stdin.read()
#print xmldoc
root = ET.fromstring(xmldoc)
# print root.attrib['lang']
lang = "unknown"
for k in root.attrib:
   if re.match(".*lang$", k):
     language = root.attrib[k]
print language
◇
```
Uses: `print` 75a.

The module-scripts depend on the existence of variable `naflang`. In most cases this is not a problem because the scripts run in a surrounding script that sets `naflang`. However, a users may occasionally run a module-script stand-alone e.g. to debug. In that case, we can read the language from the NAF, set variable `naflang`, and then run the module-script in a subshell. We assume that variable `scriptpath` contains the path of the script itself.

The macro does the following if `naflang` has not been set:

1. Save the content of standard in to a temporary file.
2. Run `langdetect` with the temporary file as input and set the `naflang` variable.
3. Run the script `$scriptpath` (i.e. itself) with the temporary file as input.
4. Remove the temporary file.
5. Exit itself with the errorcode of the sub-script that it has run.

⟨ *run in subshell when naflang is not known* 27b ⟩ ≡
```
if
  [ "$naflang" == "" ]
then
  naffile=‘mktemp -t naf.XXXXXX‘
  cat >$naffile
  naflang=‘cat $naffile | python $envbindir/langdetect.py‘
  export naflang
  cat $naffile | $scriptpath
  result=$?
  rm $naffile
  exit $result
fi
◇
```
Fragment referenced in 26b.
Uses: `naflang` 66a, `scriptpath` 26c.

⟨ *run only if language is English or Dutch* 28a ⟩ ≡

```
if
  [ ! "$naflang" == "nl" ] && [ ! "$naflang" == "en" ]
then
  exit 6
fi
```

◇

Fragment referenced in 26b.
Uses: naflang 66a.

5.5.4 Alpino

Binary versions of Alpino can be obtained from the official Alpino website of Gertjan van Noort. However, it seems that older versions are not always retained there, or the location of older versions change. Therefore we have a copy in the snapshot.

Module

⟨ *install Alpino* 28b ⟩ ≡

```
if
  [ ! $alpino_installed ]
then
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
sicstus.tar.gz
  echo "export alpino_installed=0" >> /home/huygen/projecten/nlpp/installed_modules
fi
```

◇

Fragment referenced in 20a.

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* 28c ⟩ ≡

```
export ALPINO_HOME=$modulesdir/Alpino
```

◇

Fragment referenced in 41e.
Defines: ALPINO_HOME Never used.

Remove the tarball when cleaning up:

⟨ *clean up* 28d ⟩ ≡

```
rm -rf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
sicstus.tar.gz
```

◇

Fragment defined by 12e, 13c, 28d, 72a.
Fragment referenced in 71a.

5.5.5 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* 29a ⟩ ≡
```
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 29abcd, 30abc.
Fragment referenced in 20a.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 29b ⟩ ≡
```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
◇
```
Fragment defined by 29abcd, 30abc.
Fragment referenced in 20a.
Uses: `install` 81d.

Parametersets:

⟨ *install the treetagger utility* 29c ⟩ ≡
```
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 29abcd, 30abc.
Fragment referenced in 20a.

Download everything in the target directory:

⟨ *install the treetagger utility* 29d ⟩ ≡
```
mkdir -p $modulesdir/$TREETAGDIR
cd $modulesdir/$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 29abcd, 30abc.
Fragment referenced in 20a.

Run the install-script:

⟨ *install the treetagger utility* 30a ⟩ ≡
```
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```
Fragment defined by 29abcd, 30abc.
Fragment referenced in 20a.

Make the treetagger utilities available for everybody.

⟨ *install the treetagger utility* 30b ⟩ ≡
```
chmod -R o+rx $modulesdir/$TREETAGDIR/bin
chmod -R o+rx $modulesdir/$TREETAGDIR/cmd
chmod -R o+r $modulesdir/$TREETAGDIR/doc
chmod -R o+rx $modulesdir/$TREETAGDIR/lib
◇
```
Fragment defined by 29abcd, 30abc.
Fragment referenced in 20a.

Remove the tarballs:

⟨ *install the treetagger utility* 30c ⟩ ≡
```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```
Fragment defined by 29abcd, 30abc.
Fragment referenced in 20a.

5.5.6 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 `lib` and the `bin` sub-directories of the `env` directory.

⟨ *install the ticcutils utility* 30d ⟩ ≡
```
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 31b ⟩
◇
```
Fragment referenced in 20a, 31c.

⟨ *install the timbl utility* 31a ⟩ ≡
```
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
⟨ unpack ticcutils or timbl 31b ⟩
◇
```
Fragment referenced in 20a, 31c.

⟨ *unpack ticcutils or timbl* 31b ⟩ ≡
```
SUCCES=0
ticbeldir=‘mktemp -t -d tickbel.XXXXXX‘
cd $ticbeldir
tar -xzf $snapshotsocket/t_nlpp_resources/$TARB
cd $DIR
sh ./bootstrap.sh
./configure --prefix=$envdir
make
make install
cd $piperoot
rm -rf $ticbeldir
◇
```
Fragment referenced in 30d, 31a.
Uses: `install` 81d.

When the installation has been transplanted, Timbl and Ticcutils have to be re-installed.

⟨ *re-install modules after the transplantation* 31c ⟩ ≡
```
⟨ install the ticcutils utility 30d ⟩
⟨ install the timbl utility 31a ⟩
◇
```
Fragment referenced in 82c.

5.5.7 The Boost library

Theoretically, it is possible to download a tarball with boost from it's repository and then install it. However, I did not succeed in doing this. Therefore, I ripped the installed boost from Surfsara's Hadoop installation and put it in the `env` dir.

⟨ *install boost* 31d ⟩ ≡
```
cd $envdir
tar -xzf $snapshotsocket/t_nlpp_resources/20160103_boost_1_54_bin.tgz
◇
```
Fragment referenced in 20a.

5.5.8 Spotlight

A Spotlight server occupies a lot of memory and we need two of them, one for each language. We may be lucky and have a spotlight server running somewhere. Otherwise we have to install the server ourselves.

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://localhost:2050/rest
```

Dutch server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://localhost:2
```

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

So, let us do that:

⟨ *install the Spotlight server* 32a ⟩ ≡

```
cd $envdir
tar -xzf $snapshotsocket/t_nlpp_resources/spotlightnl.tgz
cd $envdir/spotlight
⟨ get spotlight model ball (32b nl.tar.gz ) 32d ⟩
⟨ get spotlight model ball (32c en_2+2.tar.gz ) 32d ⟩
◇
```

Fragment defined by 32a, 33a.
Fragment referenced in 20a.

⟨ *get spotlight model ball* 32d ⟩ ≡

```
if
  [ -e $snapshotsocket/t_nlpp_resources/@1 ]
then
  tar -xzf $snapshotsocket/t_nlpp_resources/@1
else
  wget http://spotlight.sztaki.hu/downloads/archive/2014/@1
  tar -xzf @1
  rm @1
fi

◇
```

Fragment referenced in 32a.

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

⟨ *install the Spotlight server* 33a ⟩ ≡

```
cd $envdir/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 32a, 33a.
Fragment referenced in 20a.

The macro `check/start spotlight` does the following:

1. Check whether spotlight runs on the default spotlighthost.
2. If that is not the case, and the defaulthost is not `localhost`, check whether Spotlight runs on localhost.
3. If a running spotlightserver is still not found, start a spotlightserver on localhost.

Start Spotlight if it doesn't run already. Spotlight ought to run on localhost unless variable `spotlighthost` exists. In that case, check whether a Spotlight server can be contacted on that host. Otherwise, change `spotlighthost` to `localhost` and check whether a Spotlight server runs there. If that is not the case, start up a Spotlight server on localhost.

The following script, `check_start_spotlight`, has the following three optional arguments:

language: Default is exported variable `naflang` if it exists, or `en`.
spotlighthost: Name of a host that probably runs a Spotlightserver. Default: exported variable `spotlighthost` if it exists, or `localhost`.
spotlightport: Default: exported variable `spotlightport` if it exists or either 2020 or 2060 for English resp. Dutch.

`"../bin/check_start_spotlight"` 33b≡

```
#!/bin/bash
source /home/huygen/projecten/nlpp/env/bin/progenv
⟨ get commandline-arguments 25b ⟩
⟨ set default arguments for Spotlight 34a ⟩
◇
```

File defined by 33b, 34b.

Fill in default values when they cannot be found in exported variables nor in command-line arguments.

⟨ *set default arguments for Spotlight* 34a ⟩ ≡

```
if
  [ "$spotlighthost" == "" ]
then
  spotlighthost=130.37.53.11
fi
if
  [ "$spotlightport" == "" ]
then
  if
    [ "$naflang" == "nl" ]
  then
      spotlightport=2060
  else
      spotlightport=2020
  fi
fi
◇
```

Fragment referenced in 33b.
Uses: naflang 66a.

"../bin/check_start_spotlight" 34b≡

```
⟨ check listener on host, port (34c $spotlighthost,34d $spotlightport ) 35c ⟩
if
  [ $spotlightrunning -ne 0 ]
then
  if
    [ ! "$spotlighthost" == "localhost" ]
  then
    export spotlighthost="localhost"
    ⟨ check listener on host, port (34e $spotlighthost,34f $spotlightport ) 35c ⟩
  fi
fi
if
  [ $spotlightrunning -ne 0 ]
then
  ⟨ start the Spotlight server on localhost 37a, ... ⟩
fi
echo $spotlighthost:$spotlightport
◇
```

File defined by 33b, 34b.

⟨ *make scripts executable* 34g ⟩ ≡

```
chmod 775  ../bin/check_start_spotlight
◇
```

Fragment defined by 24d, 34g, 81b.
Fragment referenced in 81c.

Use function `check_start_spotlight` to find and exploit a running Spotlight-server or to die (with exit code 5) if no server can be found or created. The macro uses implicitly the exported variables `spotlighthost` and `spotlightport` if they exist.

⟨ *find a spotlightserver or exit* 35a ⟩ ≡

```
spothostport=‘/home/huygen/projecten/nlpp/bin/check_start_spotlight -l $naflang‘
export spotlighthost=‘echo $spothostport | gawk -F ":" ’{print $1}’‘
export spotlightport=‘echo $spothostport | gawk -F":" ’{print $2}’‘
echo "Spotlight server found on $spothostport." >&2
if
  [ "$spotlighthost" == "none" ]
then
  echo "No Spotlight-server found."
  exit 5
fi
◇
```

Fragment referenced in 44a, 56b.
Uses: `naflang` 66a, `print` 75a.

Set the port-number and the language resource for Spotlight, dependent of the language that the user gave as argument.

⟨ *get spotlight language parameters* 35b ⟩ ≡

```
if
  [ "$naflang" == "nl" ]
then
   spotlightport=2060
else
  spotlightport=2020
fi
◇
```

Fragment never referenced.
Uses: `naflang` 66a.

The following macro has a hostname and a port-number as arguments. It checks whether something in the host listens on the port and sets variable `success` accordingly:

⟨ *check listener on host, port* 35c ⟩ ≡

```
exec 6<>/dev/tcp/@1/@2 2>/dev/null
spotlightrunning=$?
exec 6<&-
exec 6>&-
◇
```

Fragment referenced in 34b, 37c.

If variable `spotlighthost` does not exist, set it to localhost. Test whether a Spotlightserver runs on `spotlighthost`. If that fails and `spotlighthost` did not point to localhost, try localhost.

If the previous attempts were not succesfull, start the spotlightserver on localhost.

If some spotlightserver has been contacted, set variable `spotlightrunning`. Otherwise exit. At the end variable `spotlighthost` ought to contain the address of the Spotlight-host.

⟨ *try to obtain a running spotlightserver* 36a ⟩ ≡

```
⟨ test whether spotlighthost runs (36b $spotlighthost ) 36e ⟩
if
  [ ! $spotlightrunning ]
then
  if
    [ "$spotlighthost" != "localhost" ]
  then
    export spotlighthost=localhost
    ⟨ test whether spotlighthost runs (36c $spotlighthost ) 36e ⟩
  fi
fi
if
  [ ! $spotlightrunning ]
then
  ⟨ start the Spotlight server on localhost 37a, ... ⟩
  ⟨ test whether spotlighthost runs (36d $spotlighthost ) 36e ⟩
fi
if
  [ ! $spotlightrunning ]
then
  echo "Cannot start spotlight"
  exit 4
fi
◇
```

Fragment never referenced.

Test whether the Spotlightserver runs on a given host. The “spotlight-test” does not really test Spotlight, but it tests whether something is listening on the port and host where we expect Spotlight. I found the test-construction that is used here on Stackoverflow. If the test is positive, set variable `spotlightrunning` to 0. Otherwise, unset that variable.

⟨ *test whether spotlighthost runs* 36e ⟩ ≡

```
exec 6<>/dev/tcp/@1/2060
if
  [ $? -eq 0 ]
then
  export spotlightrunning=0
else
  spotlightrunning=
fi
exec 6<&-
exec 6>&-
◇
```

Fragment referenced in 36a.

When trying to start the Spotlight-server on localhost, take care that only one process does this. So we do this:

1. Try to acquire a lock without waiting for it.
2. If we got the lock, run the Spotlight java program in background.
3. If we got the lock, release it.
4. If we did not get the lock, wait for the lock to be released by the process that started the spotlight-server.

But first, we specify the resources for the Spotlight-server.

⟨ *start the Spotlight server on localhost* 37a ⟩ ≡

```
if
  [ "$naflang" == "nl" ]
then
  spotresource="nl"
else
  spotresource="en_2+2"
fi
spotlightjar=dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar
◇
```

Fragment defined by 37ab.
Fragment referenced in 34b, 36a.
Uses: `naflang` 66a.

⟨ *start the Spotlight server on localhost* 37b ⟩ ≡

```
local oldd=‘pwd‘
cd /home/huygen/projecten/nlpp/env/spotlight
$envbindir/sematree acquire spotlock 0
gotit=$?
if
  [ $gotit == 0 ]
then
  java -jar -Xmx8g $spotlightjar $spotresource \
       http://localhost:$spotlightport/rest  &
  ⟨ wait until the spotlight server is up or faulty 37c ⟩
  $envbindir/sematree release spotlock
else
  ⟨ wait until the spotlight server is up or faulty 37c ⟩
fi
cd $oldd
◇
```

Fragment defined by 37ab.
Fragment referenced in 34b, 36a.

When the Sportlight server has been started, it takes op to a minute until it really listens on its port. When there is something wrong, it will never listen, of course. Therefore, we give it three minutes. If after that time still nothing listens, we set `spotlighthost` to `none`, indicating that something has gone wrong.

⟨ *wait until the spotlight server is up or faulty* 37c ⟩ ≡

```
trial=0
maxtrials=12
while
  trial=$((trial+1))
  ⟨ check listener on host, port (37d $spotlighthost,37e $spotlightport ) 35c ⟩
  [ $spotlightrunning -ne 0 ] && [ $trial -le $maxtrials ]
do
  sleep 10
done
if
  [ $spotlightrunning -ne 0 ]
then
 export spotlighthost="none"
fi
◇
```

Fragment referenced in 37b.

Start the Spotlight if it is not already running. First find out what the host is on which we may expect to find a listening Spotlight.

Variable `spotlighthost` contains the address of the host where we expect to find Spotlight. If the expectation does not come true, and the Spotlighthost was not localhost, test whether Spotlight can be found on localhost. If the spotlight-server cannot be found, start it up on localhost.

5.5.9 VUA-pylib

Module VUA-pylib is needed for the opinion-miner. Install it in the Python library

⟨ *install VUA-pylib* 38a ⟩ ≡
```
cd $envdir/python
git clone https://github.com/cltl/VUA_pylib.git
```
◇

Fragment referenced in 20a.

5.5.10 SVMLight

SVMlight supplies a Support Vector Machine. It is used by the opinion-miner. SVMlight can be obtained from the site where it is documented.

Installation goes like this:

⟨ *install SVMLight* 38b ⟩ ≡
```
tempdir=‘mktemp -d -t SVMlight.XXXXXX‘
cd $tempdir
wget http://download.joachims.org/svm_light/current/svm_light.tar.gz
tar -xzf svm_light.tar.gz
make all
cp svm_classify /home/huygen/projecten/nlpp/env/bin/
cp svm_learn /home/huygen/projecten/nlpp/env/bin/
cd /home/huygen/projecten/nlpp
rm -rf $tempdir
```
◇

Fragment referenced in 20a.
Uses: `all` 70c.

5.5.11 CRFsuite

CRFsuite is an implementation of Conditional Random Fields (CRF). Module opinion-miner-deluxe needs it. It can be installed from it's sources, but I did not manage to this. Therefore, currently we use a pre-compiled ball.

⟨ *install CRFsuite* 39a ⟩ ≡

```
tempdir=‘mktemp -d -t crfsuite.XXXXXX‘
cd $tempdir
tar -xzf $snapshotsocket/t_nlpp_resources/crfsuite-0.12-x86_64.tar.gz
cd crfsuite-0.12
cp -r bin/crfsuite $envbindir/
mkdir -p $envdir/include/
cp -r include/* $envdir/include/
mkdir -p $envdir/lib/
cp -r lib/* $envdir/lib/
cd /home/huygen/projecten/nlpp
rm -rf $tempdir
◇
```

Fragment referenced in 20a.

5.6 Install modules

5.6.1 Install tokenizer

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

To install the tokenizer, we proceed as follows:

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

⟨ *install the tokenizer* 39b ⟩ ≡

```
tempdir=‘mktemp -d -t tok.XXXXXX‘
cd $tempdir
git clone https://github.com/ixa-ehu/ixa-pipe-tok.git
cd ixa-pipe-tok
git checkout 56f83ce4b61680346f15e5d4e6de6293764f7383
mvn clean package
mv target/ixa-pipe-tok-1.8.0.jar $jarsdir
cd $piperoot
rm -rf $tempdir
◇
```

Fragment referenced in 21a.

Script The script runs the tokenizerscript.

"../bin/tok" 39c≡

```
⟨ start of module-script (39d $jarsdir ) 26b ⟩
JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
java -Xmx1000m  -jar $JARFILE tok -l $naflang --inputkaf
◇
```

5.6.2 Topic analyser

Install the topic tool `ixa-pipe-topic` that is based on JEX.

Installation goes as follows:

1. Clone from Github.
2. Download JEX resources and JEX jar libraries and put them at proper places.
3. Download and run a utility, `install-to-project-repo.py`, that puts the JEX libraries in a place where Maven can find them.
4. run maven

⟨ *install the topic analyser* 40a ⟩ ≡

```
cd $modulesdir
git clone https://github.com/ialdabe/ixa-pipe-topic.git
cd ixa-pipe-topic
git checkout 40be8debb88093b426ae3520d60df60161968e27
tempdir=‘mktemp -d -t topinambour.XXXXXX‘
moddir=$modulesdir/ixa-pipe-topic
⟨ install the jex resources and libraries 40b ⟩
⟨ compile the topic-tool jar 41a ⟩
cd $modulesdir
rm -rf $tempdir
◇
```

Fragment referenced in 21a.

The two zip-balls `http://optima.jrc.it/Resources/Eurovoc/indexing/en-eurovoc-1.0.zip` and `http://optima.jrc.it/Resources/Eurovoc/indexing/nl-eurovoc-1.0.zip` contain resources in a sub-directory `resources` and jar libs in a subdirectory `jar`. The jars in the two zip-balls are identical, so the jars from one of the balls can be copied to the `lib` subdirectory of the module where the compilation-tool expects them. The `resources` directories are placed in subdirectories `en` resp. `nl` of the `jex` subdirectory of the module directory.

⟨ *install the jex resources and libraries* 40b ⟩ ≡

```
moddir=$modulesdir/ixa-pipe-topic
cd $moddir
mkdir -p jex/en
mkdir -p jex/nl
mkdir -p lib
cd $tempdir
wget http://optima.jrc.it/Resources/Eurovoc/indexing/en-eurovoc-1.0.zip
wget http://optima.jrc.it/Resources/Eurovoc/indexing/nl-eurovoc-1.0.zip
unzip -q en-eurovoc-1.0.zip
unzip -q nl-eurovoc-1.0.zip
cp -r en-eurovoc-1.0/resources $moddir/jex/en/
cp -r nl-eurovoc-1.0/resources $moddir/jex/nl/
cp -r nl-eurovoc-1.0/lib/*.jar $moddir/lib/
◇
```

Fragment referenced in 40a.

To make the jar's in the `lib` directory accessible for Maven, we use the install-to-project-repo utility. So, unpack and run this utility and finally, run Maven:

⟨ *compile the topic-tool jar* 41a ⟩ ≡
```
git clone https://github.com/carchrae/install-to-project-repo.git
cd $modulesdir/ixa-pipe-topic
python $tempdir/install-to-project-repo/install-to-project-repo.py
mvn clean install
◇
```
Fragment referenced in 40a.
Uses: `install` 81d.

Script: The topic module uses a temporary directory to store intermediate results. To tell the Java program where the temp storage is, a config file has to generated on the fly.

`"../bin/topic"` 41b≡
```
⟨ start of module-script (41c ixa-pipe-topic ) 26b ⟩
tempdir=‘mktemp -d -t jex.XXXXXX‘
mkdir $tempdir/documents
mkdir $tempdir/results
cat $MODDIR/default.prop \
    | sed 's|jex/resources|'${MODDIR}'/jex/LANG/resources|g' \
    | sed 's|jex/result|'${tempdir}'/jex/result|g' \
    | sed 's|LANG|'${naflang}'|g' \
    >$tempdir/conf.prop
java -Xmx1000m -jar $MODDIR/target/ixa-pipe-topic-1.0.3.jar -p $tempdir/conf.prop
rm -rf $tempdir
◇
```

5.6.3 Morphosyntactic parser

Module

⟨ *install the morphosyntactic parser* 41d ⟩ ≡
```
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=d5f002605d7c06545f24c84386342b79e5cb9c86
⟨ install from github 9a ⟩
cd $modulesdir/morphosyntactic_parser_nl
git checkout d5f002605d7c06545f24c84386342b79e5cb9c86
◇
```
Fragment referenced in 21a.

Script The morpho-syntactic module parses the sentences with Alpino. Alpino takes a lot of time to handle long sentences. Therefore the morpho-syntactic module has an option `-t` to set a time-out (in minutes) for sentence parsing.

`"../bin/mor"` 41e≡
```
⟨ start of module-script (41f morphosyntactic_parser_nl ) 26b ⟩
⟨ get the mor time-out parameter 42a ⟩
⟨ set alpinohome 28c ⟩
cat | python $MODDIR/core/morph_syn_parser.py $timeoutarg
◇
```

Use getopts to read the `-t` option.

⟨ *get the mor time-out parameter* 42a ⟩ ≡

```
OPTIND=1
stimeout=
timeoutarg=
while getopts "t:" opt; do
    case "$opt" in
    t)  stimeout=$OPTARG
        ;;
    esac
done
shift $((OPTIND-1))
if
  [ $stimeout ]
then
  timeoutarg="-t $stimeout"
fi
```

◇

Fragment referenced in 41e.

5.6.4 Pos tagger

In the Dutch pipeline the morpho-syntactic parser fulfills the role of Pos tagger. In the English pipeline we use the pos-tagger from EHU.

Module

⟨ *install the pos tagger* 42b ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-pos.v30.tgz
cd $modulesdir/ixa-pipe-topic
```

◇

Fragment referenced in 21a.

Script

`"../bin/pos"` 42c≡

⟨ *start of module-script* (42d `EHU-pos.v30`) 26b ⟩

```
java -Xmx1000m -jar ${MODDIR}/ixa-pipe-pos-1.4.3.jar tag -m ${MODDIR}/en-maxent-
100-c5-baseline-dict-penn.bin
```

◇

5.6.5 Constituent parser

Module

⟨ *install the constituents parser* 43a ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-parse.v30.tgz
cd $modulesdir/conspardir
chmod 775 *.jar
chmod 775 *.bin
◇
```

Fragment referenced in 21j.

Script

"../bin/constpars" 43b≡

```
⟨ start of module-script (43c EHU-parse.v30 ) 26b ⟩
java -Xmx1000m -jar ${MODDIR}/ixa-pipe-parse-1.1.1.jar parse -g sem -
m ${MODDIR}/en-parser-chunking.bin
◇
```

5.6.6 NED-reranker

Module

⟨ *install the* NED*-reranker module* 43d ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-popen-nedreranker.v30.tgz
◇
```

Fragment referenced in 21j.

Script

"../bin/nedrer" 43e≡

```
⟨ start of module-script (43f VUA-popen-nedreranker.v30 ) 26b ⟩
cd $MODDIR
python $MODDIR/domain_model.py
◇
```

5.6.7 Wikify module

Module

⟨ *install the wikify module* 43g ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-wikify.v30.tgz
◇
```

Fragment referenced in 21j.

Script The Wikify module needs DBpedia to generate “markables”.

```
"../bin/wikify" 44a≡
    ⟨ start of module-script (44b EHU-wikify.v30 ) 26b ⟩
    ⟨ find a spotlightserver or exit 35a ⟩
    cd $MODDIR
    java -Xmx1000m -jar ${MODDIR}/ixa-pipe-wikify-1.2.1.jar -s http://$spotlighthost -
    p $spotlightport
    ◇
```

5.6.8 UKB

UKB needs boost libraries and Perl version 5. For now, we consider them installed.

Module

⟨ *install the UKB module* 44c ⟩ ≡
◇
Fragment never referenced.

Script Put the path to the boost libraries in the `LD_LIBRARY_PATH` variable and then run UKB.

Note that we cannot call perl impicitly with the hashbang.

```
"../bin/ukb" 44d≡
    ⟨ start of module-script (44e EHU-ukb.v30 ) 26b ⟩
    cd $MODDIR
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$envdir/boost_1_54_0/stage/lib
    perl ${MODDIR}/naf_ukb/naf_ukb.pl -x ${MODDIR}/ukb/bin/ukb_wsd -K ${MODDIR}/wn30-
    ili_lkb/wn30g.bin64 -D ${MODDIR}/wn30-ili_lkb/wn30.lex - -- --dict_weight --
    dgraph_dfs --dgraph_rank ppr

    ◇
```

5.6.9 IMS-WSD

Module The package itself supplies an installation script that seems usable. However, today I am in a hurry and just install the module as it comes from the EHU repository.

Although the Hadoop implementation runs this module with Java 1.7, I could only run `ims+wsd` Java 1.6. Using Java 1.7 causes run-time errors "Platform not recognised" and the resulting NAF's do not contain WordNet references. So, we had to install Java 1.6.

The scripts contain explicit paths that must be corrected:

`ims/testPlain`: Explicit path to Java binary.
`path_to_ims.py`: Set variable `PATH_TO_IMS`.

⟨ *install the ims-wsd module* 45a ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-ims-wsd.v30.tgz
cd VUA-ims-wsd.v30
thisDir=‘pwd‘
echo PATH_TO_IMS = "’"$thisDir/ims"’" > path_to_ims.py
cd ims
cp testPlain.bash old.testPlain.bash
sedcommand=’s|/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0.x86_64/jre/bin/java|java|g’
cat old.testPlain.bash | sed $sedcommand >testPlain.bash
◇
```

Fragment referenced in 22a.

Script

"../bin/ewsd" 45b≡

```
⟨ start of module-script (45c VUA-ims-wsd.v30 ) 26b ⟩
⟨ set up Java 1.6 13f ⟩
#Setting the output to be ili-wn30 synsets instead of sensekeys
$MODDIR/call_ims.py -ili30
◇
```

5.6.10 SRL server

The EHU SRL-module, that we use for Enghlish documents, has been set up as a server/client system. Hence, we have to start the server before we can process something.

We don’t know in advance whether we run the pipeline for a single text or from a whole bunch of text and hence we do not know whether it is advisable that the server keeps running, occupying precious memory. Therefore, currently we just start and stop the server every time that we use it.

Module

⟨ *install the srl-server module* 45d ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-srl-server.tgz
cd EHU-srl-server
mkdir -p /home/huygen/projecten/nlpp/env/etc/pid
◇
```

Fragment referenced in 22a.

Scripts Generate three scripts: `start_eSRL`, `stop_esrl` and `eSRL`, resp. to start the SRL server, to stop it and to process a NAF file.

"../bin/start_eSRL" 45e≡

```
⟨ start of module-script (45f EHU-srl-server ) 26b ⟩
⟨ start EHU SRL server if it isn’t running 46e ⟩
◇
```

"../bin/stop_eSRL" 46a≡
⟨ *start of module-script* (46b EHU-srl-server) 26b ⟩
⟨ *stop EHU SRL server* 46f ⟩
◇

"../bin/eSRL" 46c≡
⟨ *start of module-script* (46d EHU-srl-server) 26b ⟩
```
/home/huygen/projecten/nlpp/bin/start_eSRL
java -Xmx1000m -cp $MODDIR/IXA-EHU-srl-3.0.jar ixa.srl.SRLClient en
```
◇

⟨ *start EHU SRL server if it isn't running* 46e ⟩ ≡
```
pidFile=/home/huygen/projecten/nlpp/env/etc/pid/SRLServer.pid
portInfo=$(nmap -p 5005 localhost | grep open)
if [ -z "$portInfo" ]; then
  >&2 echo "Starting srl-server as it is not runnning"
  java -Xms2500m -cp $MODDIR/IXA-EHU-srl-
3.0.jar ixa.srl.SRLServer en &> /dev/null &
  pid=$!
  echo $pid > $pidFile
  sleep 60
  >&2 echo "Server running: ${pid}"
else
 >&2 echo "Server already running.."
fi
```
◇

Fragment referenced in 45e.

⟨ *stop EHU SRL server* 46f ⟩ ≡
```
pidFile=/home/huygen/projecten/nlpp/env/etc/pid/SRLServer.pid
if
 [ -e "$pidFile" ]
then
 kill `echo $pidFile`
 rm $pidFile
fi
```
◇

Fragment referenced in 46a.

5.6.11 SRL Dutch nominals

Module

⟨ *install the srl-dutch-nominals module* 46g ⟩ ≡
```
MODNAM=srl-dutch-nominals
DIRN=vua-srl-dutch-nominal-events
GITU=https://github.com/newsreader/vua-srl-dutch-nominal-events
GITC=6115b3168978acf809916cd2da512295d109d8fb
```
⟨ *install from github* 9a ⟩
```
cd $modulesdir/vua-srl-dutch-nominal-events
```
◇

Fragment referenced in 22a.

Script

```
"../bin/srl-dutch-nominals" 47a≡
    ⟨ start of module-script (47b vua-srl-dutch-nominal-events ) 26b ⟩
    cd $MODDIR
    cat | python $MODDIR/vua-srl-dutch-additional-roles.py
    ◇
```

5.6.12 FBK-time module

Module

```
⟨ install the FBK-time module 47c ⟩ ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-time.v30.tgz
    ◇
```

Fragment referenced in 22j.

Script The script is rather complicated. I just copied it from the orignal makers, with one exception: Originally at the end of the script there was a pipe consisting of two Java programs. However, that didn't seem to work in one of the computers that we use, therefore we have split the pipe using `mytemp` as temporary storage.

"../bin/FBK-time" 48a≡
⟨ *start of module-script* (48b FBK-time.v30) 26b ⟩
```
BEGINTIME=‘date ’+%Y-%m-%dT%H:%M:%S%z’‘
YAMCHA=$MODDIR/tools
timdir=‘mktemp -d -t time.XXXXXX‘
FILETXP=$timdir/TimePro.txp
CHUNKIN=$timdir/TimePro.naf
FILEOUT=$timdir/TimeProOUT.txp
TIMEPRONORMIN=$timdir/TimeProNormIN.txp
JAVAMAXHEAP=2g
mytemp=$timdir/mytemp
result=0
cd $MODDIR
cat > $CHUNKIN

JAVACLASSPATH="lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar:lib/NAFtoTXP_v11.jar"
JAVAMODULE=eu.fbk.newsreader.naf.NAFtoTXP_v11
cat $CHUNKIN | \
 java -Xmx$JAVAMAXHEAP -cp $JAVACLASSPATH $JAVAMODULE $FILETXP chunk+entity timex
```
⟨ *stop on error* (48c Java: $JAVACLASSPATH:$JAVAMODULE) 49a ⟩
```
#echo "Saving... $FILETXP"
tail -n +4 $FILETXP | awk -f resources/english-rules > $FILEOUT
head -n +4 $FILETXP > $TIMEPRONORMIN

cat $FILEOUT | \
  $YAMCHA/yamcha-0.33/usr/local/bin/yamcha \
    -m models/tempeval3_silver-data.model \
  >> $TIMEPRONORMIN
```
⟨ *stop on error* (48d yamcha) 49a ⟩
```
JAVACLASSPATH="lib/scala-library.jar:lib/timenorm-0.9.1-SNAPSHOT.jar"
JAVACLASSPATH=$JAVACLASSPATH:"lib/threetenbp-0.8.1.jar:lib/TimeProNorm_v2.5.jar"
JAVAMODULE=eu.fbk.timePro.TimeProNormApply
cat $TIMEPRONORMIN | \
  java -Xmx$JAVAMAXHEAP -cp $JAVACLASSPATH $JAVAMODULE $FILETXP
```
⟨ *stop on error* (48e Java: $JAVACLASSPATH:$JAVAMODULE) 49a ⟩
```
rm $FILEOUT
rm $TIMEPRONORMIN

JAVACLASSPATH="lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar:lib/NAFtoTXP_v11.jar"
JAVAMODULE=eu.fbk.newsreader.naf.NAFtoTXP_v11
cat $CHUNKIN | java -Xmx$JAVAMAXHEAP -
cp $JAVACLASSPATH $JAVAMODULE $FILEOUT chunk+morpho+timex+event eval
```
⟨ *stop on error* (48f Java: $JAVACLASSPATH:$JAVAMODULE) 49a ⟩
```
JAVACP1="lib/TXPtoNAF_v5.jar:lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar"
JAVAMOD1=eu.fbk.newsreader.naf.TXPtoNAF_v4
JAVACP2="lib/kaflib-naf-1.1.9.jar:lib/jdom-2.0.5.jar:lib/TimeProEmptyTimex_v2.jar"
JAVAMOD2=eu.fbk.timepro.TimeProEmptyTimex
java -Xmx$JAVAMAXHEAP  -Dfile.encoding=UTF8 -
cp $JAVACP1 $JAVAMOD1 $CHUNKIN $FILETXP "$BEGINTIME" TIMEX3 > $mytemp
cat $mytemp | java -Xmx$JAVAMAXHEAP -Dfile.encoding=UTF8 -
cp $JAVACP2 $JAVAMOD2 $FILEOUT
```
⟨ *stop on error* (48g Java: $JAVACLASSPATH:$JAVAMODULE) 49a ⟩
```
rm $FILETXP
rm $CHUNKIN
rm -rf $timdir
```
◇

When one of the programs in the script fail, stop processing. Pass the error-code and write a message to locate the failing program. Remove the temporary directory. However, there is a problem. One of the java programs always results with result-code 1.

⟨ *stop on error* 49a ⟩ ≡
```
result=$?
if
  [ $result -ne 0 ]
then
  cd $MODDIR
  echo Error: @1 >&2
  rm -rf $timdir
  exit $result
fi
```
◇

Fragment referenced in 48a.

5.6.13 FBK-temprel module

Module

⟨ *install the FBK-temprel module* 49b ⟩ ≡
```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-temprel.v30.tgz
⟨ repair FBK-*rel's run.sh.hadoop (49c FBK-temprel.v30 ) 49d ⟩
```
◇

Fragment referenced in 22j.

Script run.sh.hadoop seems to be obsolete in the original tarball:

1. The class-path argument in one of the Java statement refers to an obsolete jar (`kaflib-naf-1.1.8` instead of `kaflib-naf-1.1.9`)
2. Another class-path argument refers to `PredicateTimeAnchor_tlink.jar` instead of `PredicateTimeAnchor.ja`
3. A "sh" statement is used. The problem is, that in Ubuntu `/bin/sh` points to `bin/dash` and the script (`temprel-pipeline-per-file-NWR.sh`) does not seem to be compatible with `dash`.

Therefore, we need to repair the script. We will need to repair the script in the `FBK-causalrel` module in a similar way, and therefore provide the module-directory as argument.

⟨ *repair FBK-*rel's run.sh.hadoop* 49d ⟩ ≡
```
cd $modulesdir/@1
mv run.sh.hadoop old.run.sh.hadoop
cat old.run.sh.hadoop | \
  sed s/kaflib-naf-1.1.8/kaflib-naf-1.1.9/g | \
  sed s/TimeAnchor_tlink.jar/TimeAnchor.jar/g | \
  sed "s/sh temprel/bash temprel/g" | \
  sed "s/java /java -Xmx2g /g"  \
>run.sh.hadoop
chmod 775 run.sh.hadoop
```
◇

Fragment referenced in 49b, 50c.

Script The original run script seems to not only read the input naf from standard in, but also to obtain the input naf as a file that an argument points to. This constructions makes the pipeline complicated, therefore, we generate the naf file within the script.

The original script generates temporary files in the `temp` directory of the host-computer, and prefixes the names of the temporary files with a random number to prevent confusion between tempfiles of different instances of this module. We generate a temp-directory per instance.

```
"../bin/FBK-temprel" 50a≡
    ⟨ start of module-script (50b FBK-temprel.v30 ) 26b ⟩
    cd $MODDIR
    scratchDir=‘mktemp -d -t temprel.XXXXXX‘
    cat >$scratchDir/in.naf
    ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
    rm -rf $scratchDir

    ◇
```

5.6.14 FBK-causalrel module

Module

```
⟨ install the FBK-causalrel module 50c ⟩ ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-causalrel.v30.tgz
    ⟨ repair FBK-*rel's run.sh.hadoop (50d FBK-causalrel.v30 ) 49d ⟩
    ◇
```

Fragment referenced in 22j.

Like in `FBK-temprel`, script run.sh.hadoop seems not to work out of the box:

1. The class-path argument in one of the Java statement refers to an obsolete jar (`kaflib-naf-1.1.8` instead of `kaflib-naf-1.1.9`)
2. A "sh" statement is used. The problem is, that in Ubuntu `/bin/sh` points to `bin/dash` and the script (`temprel-pipeline-per-file-NWR.sh`) does not seem to be compatible with `dash`.

Therefore, we need to repair that script like we did in `FBK-temprel`.

```
⟨ repair causalrel's run.sh.hadoop 50e ⟩ ≡
    cd $modulesdir/FBK-causalrel.v30
    mv run.sh.hadoop old.run.sh.hadoop
    cat old.run.sh.hadoop | \
      sed s/kaflib-naf-1.1.8/kaflib-naf-1.1.9/g | \
      sed s/TimeAnchor_tlink.jar/TimeAnchor.jar/g | \
      sed s/sh temprel/bash temprel/g | \
    >run.sh.hadoop
    chmod 775 run.sh.hadoop
    ◇
```

Fragment never referenced.

Script

```
"../bin/FBK-causalrel" 51a≡
        ⟨ start of module-script (51b FBK-causalrel.v30 ) 26b ⟩
        cd $MODDIR
        scratchDir=‘mktemp -d -t causalrel.XXXXXX‘
        cat >$scratchDir/in.naf
        ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
        rm -rf $scratchDir
        ◇
```

5.6.15 Factuality module

Module

```
⟨ install the factuality module 51c ⟩ ≡
        cd $modulesdir
        tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-factuality.v30.tgz
        ◇
```

Fragment referenced in 22j.

Script

```
"../bin/factuality" 51d≡
        ⟨ start of module-script (51e VUA-factuality.v30 ) 26b ⟩
        cd $MODDIR
        #local settings to prevent perl from complaining
        export LANGUAGE=en_US.UTF-8
        export LANG=en_US.UTF-8
        export LC_ALL=en_US.UTF-8

        rootDir=${MODDIR}
        tmpDir=$(mktemp -d -t factuality.XXXXXX)

        export PATH=$PATH:${rootDir}:.
        # ex-
        port LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${rootDir}/../opt/lib/:${rootDir}/../opt/boost_1_54_0/stage/lib
        export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/huygen/projecten/nlpp/env/lib/

        #mkdir -p ${scratchDir}/test

        python ${rootDir}/vua_factuality_naf_wrapper.py -
        t /home/huygen/projecten/nlpp/env/bin/timbl -p ${rootDir} ${tmpDir}/

        ◇
```

5.6.16 Nominal coreference-base

The source of this module in Github (https://github.com/opener-project/coreference-base.git) does not seem to work well with NAF. Therefore, we use the version from the official English pipeline, that we find in the snapshot.

Module

⟨ *install coreference-base* 52a ⟩ ≡

```
cd $modulesdir
tar -xzf /home/huygen/projecten/t_nlpp_resources/20151220_EHU-corefgraph.v30.tgz
◇
```

Fragment referenced in 23a.

Script

"../bin/coreference-base" 52b≡

```
⟨ start of module-script (52c EHU-corefgraph.v30 ) 26b ⟩
cd $MODDIR/corefgraph
cat | python -m corefgraph.process.file --reader NAF --writer NAF

◇
```

5.6.17 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* 52d ⟩ ≡

```
⟨ compile the nerc jar 52e ⟩
⟨ get the nerc models 53b ⟩

◇
```

Fragment referenced in 21j.

The nerc module is a Java program that is contained in a jar. Put 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* 52e ⟩ ≡

```
TEMPDIR=‘mktemp -d -t nerc.XXXXXX‘
cd $TEMPDIR
git clone https://github.com/ixa-ehu/ixa-pipe-nerc
cd ixa-pipe-nerc/
git checkout ca02c931bc0b200ccdb8b5795a7552e4cc0d4802
mvn clean package
mv target/ixa-pipe-nerc-1.5.4.jar $jarsdir/
cd $nuwebdir
rm -rf $TEMPDIR
◇
```

Fragment referenced in 52d.

The current version of the pipeline uses the following models, that have been made available by Rodrigo Agerri on december 15, 2015.

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 `/m4_nerc_nl_dir/nerc_models/nl` of the snapshot.

The model for English can be found in the newsreader-repository.

Choose a model dependent of the language.

⟨ *select language-dependent features* 53a ⟩ ≡

```
if
  [ "$naflang" == "nl" ]
then
  export nercmodel=nl/nl-clusters-conll02.bin
else
  export nercmodel=en/en-newsreader-clusters-3-class-muc7-conll03-ontonotes-4.0.bin
fi
◇
```

Fragment never referenced.
Uses: naflang 66a.

The tarball 20160301_nerc_models.tgz contains in subdirectories nl and en a dutch resp. an english nerc-model. They have been randomly selected from a number of models that are available in http://ixa2.si.ehu.es/ixa-pipes/models/nerc-models-1.5.4.tgz.

⟨ *get the nerc models* 53b ⟩ ≡

```
cd $modulesdir
tar -p -xzf /home/huygen/projecten/t_nlpp_resources/20160301_nerc_models.tgz
◇
```

Fragment referenced in 52d.

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

"../bin/nerc_conll02" 53c≡

```
⟨ start of module-script (53d m4_nerc_nl_dir ) 26b ⟩
JAR=$jarsdir/ixa-pipe-nerc-1.5.4.jar
MODEL=nl-clusters-conll02.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nl/$MODEL
◇
```

"../bin/nerc" 53e≡

```
⟨ start of module-script (53f nerc-models ) 26b ⟩
JAR=$jarsdir/ixa-pipe-nerc-1.5.4.jar
if
  [ "$naflang" == "nl" ]
then
  nercmodel=$modulesdir/nerc_models/nl/nl-6-class-clusters-sonar.bin
else
  nercmodel=$modulesdir/nerc_models/en/en-best-clusters-conll03.bin
fi
java -Xmx1500m -jar $JAR tag -m $nercmodel
◇
```

5.6.18 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* 54a ⟩ ≡

```
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=030043903b42f77cd20a9b2443de137e2efe8513
⟨ install from github 9a ⟩
cd $modulesdir/svm_wsd
⟨ install svm lib 54b ⟩
⟨ download svm models 54c ⟩


◇
```

Fragment referenced in 23a.

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

⟨ *install svm lib* 54b ⟩ ≡

```
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 54a.

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

⟨ *download svm models* 54c ⟩ ≡

```
cd $modulesdir/svm_wsd
#tar -xzf $pipesocket/m4_wsd_snapball
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 54a.

Script

"../bin/wsd" 54d≡

```
⟨ start of module-script (54e svm_wsd ) 26b ⟩
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $MODDIR/$WSDSCRIPT --naf -ref odwnSY
◇
```

5.6.19 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* 55a ⟩ ≡
```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/lu2synset.tgz
◇
```
Fragment referenced in 23h.

Script

"../bin/lu2synset" 55b≡
```
⟨ start of module-script (55c lexicalunitconvertor ) 26b ⟩
JAVALIBDIR=$MODDIR/lib
RESOURCESDIR=$MODDIR/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
◇
```

5.6.20 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 NED should 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* 55d ⟩ ≡
```
⟨ put spotlight jar in the Maven repository 56a ⟩
MODNAM=ned
DIRN=ixa-pipe-ned
GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
⟨ install from github 9a ⟩
cd $modulesdir/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 $jarsdir/
◇
```
Fragment referenced in 21j.

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* 56a ⟩ ≡

```
echo Put Spotlight jar in the Maven repository.
tempdir=‘mktemp -d -t simplespot.XXXXXX‘
cd $tempdir
wget http://spotlight.sztaki.hu/downloads/archive/2014/dbpedia-spotlight-0.7.jar
wget http://spotlight.sztaki.hu/downloads/archive/2014/nl.tar.gz
tar -xzf nl.tar.gz
MVN_SPOTLIGHT_OPTIONS="-Dfile=dbpedia-spotlight-0.7.jar"
MVN_SPOTLIGHT_OPTIONS="$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 55d.
Uses: `install` 81d.

Script NED needs to contact a Spotlight-server.

`"../bin/ned"` 56b≡

```
⟨ start of module-script (56c ) 26b ⟩
ROOT=$piperoot
JARDIR=$jarsdir
⟨ find a spotlightserver or exit 35a ⟩
cat | java -Xmx1000m -jar $jarsdir/ixa-pipe-ned-1.1.1.jar -
H http://$spotlighthost -p $spotlightport -e candidates -
i $envdir/spotlight/wikipedia-db -n nlEn
◇
```

5.6.21 Ontotagger, Framenet-SRL and nominal events

The three modules ontotagger (aka “predicatematrix”), Framenet-SRL and nominal event detection are based on the same software packages and resources. The three modules need the same jar `ontotagger-1.0-jar-with-dependencies.jar`, they need resources from the `cltl/vua_resources` Github repository and they are going to execute a script that resides in the scripts directory of the `cltl/OntoTagger` repository. So, what we have to do is:

1. Install from the `cltl/OntoTagger` repository.
2. Create the jar and put it in an appropriate place.
3. install from the `cltl\vua-resources` repository.
4. generate a script fot each of the modules.

In fact, items 2 and 3 are performed by script `install.sh` from the OntoTagger repository.

Modules

⟨ *install the ontotagger repository* 57a ⟩ ≡

```
cd $modulesdir
git clone https://github.com/cltl/OntoTagger.git
cd OntoTagger
git checkout 9ea03d73eef1c9f4c85a0f05bc8137149e51335c
chmod 775 ./install.sh
./install.sh
cd $piperoot
◇
```

Fragment referenced in 23a.
Uses: `install` 81d.

Scripts The "onto" (predicatematrix) script:

`"../bin/onto"` 57b≡

```
⟨ start of module-script (57c OntoTagger ) 26b ⟩
cd $MODDIR/scripts
cat | $MODDIR/scripts/predicate-matrix-tagger.sh

◇
```

The "Framenet SRL" script:

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

`"../bin/framesrl"` 57d≡

```
⟨ start of module-script (57e OntoTagger ) 26b ⟩
cd $MODDIR/scripts
cat | $MODDIR/scripts/srl-framenet-
tagger.sh | gawk '/^frameMap.size()/ {next}; {print}'

◇
```

The "nomevent" script:

`"../bin/nomevent"` 57f≡

```
⟨ start of module-script (57g OntoTagger ) 26b ⟩
cd $MODDIR/scripts
cat | $MODDIR/scripts/nominal-events.sh
◇
```

5.6.22 Heideltime

Module The code for Heideltime can be found in Github. However, we use a compiled Heideltime Jar, compiled by Antske Fokkens, because some bugs have been repaired in that version.

Use Heideltime via a wrapper, `ixa-pipe-time`, obtained from Github.

Heideltime uses treetagger. It expects to find the location of treetagger in a variable `TreetaggerHome` in config-file `config.props`.

⟨ *install the heideltime module* 58a ⟩ ≡
```
moduledir=/home/huygen/projecten/nlpp/modules/ixa-pipe-time
⟨ clone the heideltime wrapper 58b ⟩
⟨ put Antske's material in the heideltime wrapper 58d ⟩
⟨ compile the heideltime wrapper 58e ⟩
◇
```
Fragment referenced in 23h.

⟨ *clone the heideltime wrapper* 58b ⟩ ≡
```
MODNAM=heideltime
DIRN=ixa-pipe-time
GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
GITC=da4604a7b33975e977017440cbc10f7d59917ddf
⟨ install from github (58c ixa-pipe-time ) 9a ⟩
mkdir $moduledir/lib
◇
```
Fragment referenced in 58a.

In the wrapper we need the following extra material:

- A debugged version of the Heidelberg jar.
- A configuration file `config.props`, although it does not seem to be actually used.
- Another configuration file: `alpino-to-treetagger.csv`

The extra material has been provided by Antske Fokkens.

⟨ *put Antske's material in the heideltime wrapper* 58d ⟩ ≡
```
cd $modulesdir/$DIRN
tar -xzf $snapshotsocket/t_nlpp_resources/20151123_antske_heideltime_stuff.tgz
mv antske_heideltime_stuff/de.unihd.dbs.heideltime.standalone.jar lib/
mv antske_heideltime_stuff/config.props .
mv antske_heideltime_stuff/alpino-to-treetagger.csv .
rm -rf antske_heideltime_stuff
◇
```
Fragment referenced in 58a.

Compile the Heideltime wrapper according to the instruction on Github.

⟨ *compile the heideltime wrapper* 58e ⟩ ≡
```
⟨ get jvntextpro-2.0.jar 58f ⟩
⟨ activate the install-to-project-repo utility 59a ⟩
cd /home/huygen/projecten/nlpp/modules/$DIRN
mvn clean install
◇
```
Fragment referenced in 58a.
Uses: `install` 81d.

⟨ *get jvntextpro-2.0.jar* 58f ⟩ ≡
```
cd  /home/huygen/projecten/nlpp/modules/$DIRN/lib
wget http://ixa2.si.ehu.es/%7Ejibalari/jvntextpro-2.0.jar
◇
```
Fragment referenced in 58e.

Script `install-to-project-repo.py` generates a library in subdirectory `repo` and copies the jars that it finds in the `lib` subdirectory in this repo in such a way that Maven finds it there. Somewhere in the `install-to-project.py` ... `mvn` process the jars are copied in your local repository (`~/.m2`) too. As a result, only a Maven Guru understands precisely where Maven obtains its jar from and the best thing to do is to empty the `repo` subdirectory and the local repository before (re-) applying `install-to-project-repo.py`.

⟨ *activate the install-to-project-repo utility* 59a ⟩ ≡
```
⟨ remove outdated heideltime jars 59b ⟩
cd /home/huygen/projecten/nlpp/modules/$DIRN/
git clone git@github.com:carchrae/install-to-project-repo.git
mv install-to-project-repo/install-to-project-repo.py .
rm -rf install-to-project-repo
python ./install-to-project-repo.py
◇
```
Fragment referenced in 58e.
Uses: `install` 81d.

⟨ *remove outdated heideltime jars* 59b ⟩ ≡
```
rm -rf /home/huygen/projecten/nlpp/modules/$DIRN/repo
mkdir -p /home/huygen/projecten/nlpp/modules/$DIRN/repo/local
rm -rf $HOME/.m2/repository/local/de.unihd.dbs.heideltime.standalone
rm -rf $HOME/.m2/repository/local/jvntextpro-2.0
◇
```
Fragment referenced in 59a.

Script

`"../bin/heideltime"` 59c≡
```
⟨ start of module-script (59d ixa-pipe-time ) 26b ⟩
MODDIR=$modulesdir/ixa-pipe-time
cd $MODDIR
iconv -t utf-8//IGNORE | java -Xmx1000m -jar target/ixa.pipe.time.jar -m alpino-to-
treetagger.csv -c config.props
◇
```

5.6.23 Semantic Role labelling

Module

⟨ *install the srl module* 59e ⟩ ≡
```
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=675d22d361289ede23df11dcdb17195f008c54bf
⟨ install from github 9a ⟩
◇
```
Fragment referenced in 23h.

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" 60a≡
⟨ *start of module-script* (60b vua-srl-nl) 26b ⟩
```
MODDIR=$modulesdir/vua-srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $MODDIR
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
```
◇

File defined by 60acdef.

Create a feature-vector.

"../bin/srl" 60c≡
```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
```
◇

File defined by 60acdef.

Run the trained model on the feature-vector.

"../bin/srl" 60d≡
```
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 60acdef.

Insert the SRL values into the NAF file.

"../bin/srl" 60e≡
```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
```
◇

File defined by 60acdef.

Clean up.

"../bin/srl" 60f≡
```
rm -rf $TEMPDIR
```
◇

File defined by 60acdef.

5.6.24 SRL postprocessing

In addition to the Semantic Role Labeling there is hack that finds additional semantic roles.

Module Get the module from Github. Note that this module needs rdflib

⟨ *install python packages* 61a ⟩ ≡
```
pip install rdflib
```
◇

Fragment defined by 17a, 61a.
Fragment referenced in 14a.
Defines: `rdflib` Never used.
Uses: `install` 81d.

⟨ *install the post-SRL module* 61b ⟩ ≡
```
cd $modulesdir
if
  [ -d vua-srl-postprocess ]
then
  cd vua-srl-postprocess
  git pull
else
  git clone https://github.com/newsreader/vua-srl-postprocess.git
  cd vua-srl-postprocess
fi
```
◇

Fragment referenced in 23q.

Script

`"../bin/postsrl"` 61c≡
```
⟨ start of module-script (61d vua-srl-postprocess ) 26b ⟩
cd $MODDIR
tempdir=‘mktemp -d -t postsrl.XXXXX‘
cat >$tempdir/infile
python $MODDIR/main.py -i $tempdir/infile -o $tempdir/outfile
cat $tempdir/outfile
rm -rf $tempdir
```
◇

5.6.25 Event coreference

The event-coreference module is language-independent. Although the version in the EHU-repo is 3.0, the version 2.0 used in this pipeline seems to be more recent, so we will use that.

Module Install the module from the snapshot.

⟨ *install the event-coreference module* 61e ⟩ ≡
```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151217_vua-eventcoreference_v2.tgz
cd vua-eventcoreference_v2
cp lib/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar $jarsdir
```
◇

Fragment referenced in 23h.

Script

"../bin/evcoref" 62a≡
⟨ *start of module-script* (62b vua-eventcoreference_v2) 26b ⟩
```
RESOURCESDIR=$MODDIR/resources
JARFILE=$jarsdir/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar

if
  [  "$naflang" == 'nl' ]
then
  lang_resource="odwn_orbn_gwg-LMF_1.3.xml"
else
  lang_resource="wneng-30.lmf.xml.xpos"
fi

JAVAMODULE=eu.newsreader.eventcoreference.naf.EventCorefWordnetSim
JAVAOPTIONS="--method leacock-chodorow"
JAVAOPTIONS="$JAVAOPTIONS  --wn-lmf $RESOURCESDIR/$lang_resource"
JAVAOPTIONS="$JAVAOPTIONS  --sim 2.0"
JAVAOPTIONS="$JAVAOPTIONS  --wsd 0.8"
JAVAOPTIONS="$JAVAOPTIONS  --
relations XPOS_NEAR_SYNONYM#HAS_HYPERONYM#HAS_XPOS_HYPERONYM#event"


java -Xmx812m -cp $JARFILE $JAVAMODULE  $JAVAOPTIONS

◇
```

5.6.26 Dbpedia-ner

Dbpedia-ner finds more named entities thatn NER, because it checks DBpedia for the candidate NE-'s.

Module

⟨ *install the dbpedia-ner module* 62c ⟩ ≡
```
MODNAM=dbpedia_ner
DIRN=dbpedia_ner
GITU=https://github.com/PaulHuygen/dbpedia_ner.git
GITC=ab1dcbd860f0ff29bc979f646dc382122a101fc2
```
⟨ *install from github* 9a ⟩
```
◇
```
Fragment referenced in 23q.

Script The main part of the module is a Python script. The `README.md` file of the Github repo lists the options that can be applied. One of the options is about the URL of the Spotlight server.

"../bin/dbpner" 62d≡
⟨ *start of module-script* (62e dbpedia_ner) 26b ⟩
```
cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
url http://$spotlighthost:2060/rest/candidates
◇
```

5.6.27 Opinion miner

Get `opinion-miner_deluxePP` from Github.

Module Install the module from Github.

⟨ *install the opinion-miner* 63a ⟩ ≡
```
MODNAM=opinion_miner_deluxePP
DIRN=opinion_miner_deluxePP
GITU=https://github.com/rubenIzquierdo/opinion_miner_deluxePP
GITC=5f46af89f139080ae030abe70a540f693ac4676b
⟨ install from github 9a ⟩
◇
```
Fragment defined by 63abc.
Fragment referenced in 24a.

The module contains a script `install_me.sh` that we will follow here. First install the CRF module that comes with the opinion-miner:

⟨ *install the opinion-miner* 63b ⟩ ≡
```
moduledir=$modulesdir/opinion_miner_deluxePP
#Install CRF++
crfdir=‘mktemp -d -t crf.XXXXXX‘
cd $crfdir
tar xzf $moduledir/crf_lib/CRF++-0.58.tar.gz
cd CRF++-0.58
./configure --prefix=$envdir
make
make install
echo "PATH_TO_CRF_TEST='$envbindir/crf_test'" > $moduledir/path_crf.py
cd $moduledir
rm -rf $crfdir
◇
```
Fragment defined by 63abc.
Fragment referenced in 24a.
Uses: `install` 81d.

Next, download the trained models.

⟨ *install the opinion-miner* 63c ⟩ ≡
```
##Download the models
echo Downloading the trained models.
tar -xzf $snapshotsocket/t_nlpp_resources/models_opinion_miner_deluxePP.tgz
◇
```
Fragment defined by 63abc.
Fragment referenced in 24a.

Script

`"../bin/opinimin"` 63d≡
```
⟨ start of module-script (63e opinion_miner_deluxePP ) 26b ⟩
cd $MODDIR
python tag_file.py -d hotel
◇
```

6 Utilities

6.1 Run-script and test-script

The script `nlpp` reads a NAF document from standard in and produces an annotated NAF on standard out. The script `test` annotates either a test-document that resides in the nuweb directory or a user-provided document and leaves the intermediate results in its working directory `nlpp/test`, so that, in case of problems, it is easy traceable what went wrong.

The annotion process involves a sequence in which an NLP module reads a file that contains the output from a previous module (or the input NAF file), processes it and writes the result in another file.

The following function, `runmodule`, performs the action of a single module in the sequence. It needs three arguments: 1) the name of the NAF file that the previous module produced or the input file; 2) the name of thje script that runs the module and 3) the name of the output NAF.

The function uses variable `moduleresult` to decide whether it is really going to annotate. If this variable is "false" (i.e., not equal to zero), this means that one of the previous modules failed, and it is of no use to process the input file. In that case, the function leaves `moderesult` as it is and does not process the input-file. Otherwise, it will process the input-file and it sets `moduleresult` to the result of the processing module.

⟨ *function to run a module* 64 ⟩ ≡

```
export moduleresult=0

function runmodule {
  local infile=$1
  local modulecommand=$BIND/$2
  local outfile=$3
  if
    [ $moduleresult -eq 0 ]
  then
    cat $infile | $modulecommand > $outfile
    moduleresult=$?
    if
      [ $moduleresult -gt 0 ]
    then
      failmodule=$modulecommand
      echo "Failed: module $modulecommand; result $moduleresult" >&2
      exit $moduleresult
    else
       echo "Completed: module $modulecommand; result $moduleresult" >&2
    fi
  fi
}

◇
```

Fragment referenced in 67ab.
Defines: `BIND` 67c, `moduleresult` 67ab, `runmodule` 65ab, 66a.

Note: that variable `BIND` has to be defined prior to using this function.

Use the function to annotate a NAF file that `infile` points to and write the result in a file that `outfile` points to:

⟨ *annotate dutch document* 65a ⟩ ≡

```
runmodule $infile    tok                    tok.naf
runmodule tok.naf     topic                  top.naf
runmodule top.naf   mor                     mor.naf
runmodule mor.naf     nerc                   nerc.naf
runmodule nerc.naf    wsd                    wsd.naf
runmodule wsd.naf     ned                    ned.naf
runmodule ned.naf     heideltime             times.naf
runmodule times.naf   onto                   onto.naf
runmodule onto.naf    srl                    srl.naf
runmodule srl.naf     nomevent       nomev.naf
runmodule nomev.naf   srl-dutch-nominals     psrl.naf
runmodule psrl.naf    framesrl       fsrl.naf
runmodule fsrl.naf    opinimin          opin.naf
runmodule opin.naf    evcoref        $outfile
◇
```

Fragment never referenced.
Uses: `runmodule` 64.

Similar for an English naf:

⟨ *annotate english document* 65b ⟩ ≡

```
runmodule $infile    tok                    tok.naf
runmodule tok.naf     topic                  top.naf
runmodule top.naf     pos                    pos.naf
runmodule pos.naf     constpars              consp.naf
runmodule consp.naf   nerc                   nerc.naf
runmodule nerc.naf    ned                    ned.naf
runmodule ned.naf     nedrer                 nedr.naf
runmodule nedr.naf    wikify                 wikif.naf
runmodule wikif.naf   ukb                    ukb.naf
runmodule ukb.naf     ewsd                   ewsd.naf
runmodule ewsd.naf    coreference-base       coref.naf
runmodule coref.naf   eSRL                   esrl.naf
runmodule esrl.naf    FBK-time               time.naf
runmodule time.naf    FBK-temprel            trel.naf
runmodule trel.naf    FBK-causalrel          crel.naf
runmodule crel.naf    evcoref                ecrf.naf
runmodule ecrf.naf    factuality             fact.naf
runmodule fact.naf    opinimin          $outfile
◇
```

Fragment referenced in 66a.
Uses: `runmodule` 64.

Determine the language and select one of the above macro's to annotate the document. In fact, consider the document as an English document unless `naflang` is "nl"

⟨ *annotate* 66a ⟩ ≡

```
naflang=‘cat $infile | /home/huygen/projecten/nlpp/env/bin/langdetect.py‘
export naflang
if
 [ "$naflang" == "nl" ]
then
runmodule $infile    tok                     tok.naf
runmodule tok.naf     topic                   top.naf
runmodule top.naf   mor                     mor.naf
runmodule mor.naf     nerc                    nerc.naf
runmodule nerc.naf    wsd                     wsd.naf
runmodule wsd.naf     ned                     ned.naf
runmodule ned.naf     heideltime              times.naf
runmodule times.naf   onto                    onto.naf
runmodule onto.naf    srl                     srl.naf
runmodule srl.naf     nomevent       nomev.naf
runmodule nomev.naf   srl-dutch-nominals      psrl.naf
runmodule psrl.naf    framesrl       fsrl.naf
runmodule fsrl.naf    opinimin          opin.naf
runmodule opin.naf    evcoref        $outfile
else
  ⟨ annotate english document 65b ⟩
fi
◇
```

Fragment referenced in 67ab.
Defines: `naflang` 25b, 27b, 28a, 34a, 35ab, 37a, 39c, 41b, 53ae, 62a.
Uses: `runmodule` 64.

Use the above “annotate” macro in a test script and in a run script. The scripts set a working directory and put the input-file in it, and then annotate it.

The test-script uses a special test-directory and leaves it behind when it is finished. If the user specified a language, the script copies a NAF testfile from the nuweb directory as input-file. Otherwise, the script expexts the test-directory to be present, with an input-file (named `in.naf`) in it.

⟨ *get a testfile or die* 66b ⟩ ≡

```
cd $workdir
if
  [ "$1" == "en" ]
then
  cp $ROOT/nuweb/test.en.in.naf $infile
else
  if
    [ "$1" == "nl" ]
  then
    cp $ROOT/nuweb/test.nl.in.naf  $infile
  fi
fi
if
  [ ! -e $infile ]
then
  echo "Please supply test-file $workdir/$infile or specify language"
  exit 4
fi
◇
```

Fragment referenced in 67a.
Uses: `nuweb` 77b.

This is the test-script:

"../bin/test" 67a≡
```
#!/bin/bash
oldd=‘pwd‘
⟨ set variables for test/run script 67c ⟩
workdir=$piperoot/test
mkdir -p $workdir
cd $workdir
⟨ get a testfile or die 66b ⟩
⟨ function to run a module 64 ⟩
⟨ annotate 66a ⟩
if
  [ $moduleresult -eq 0 ]
then
  echo Test succeeded.
else
  echo Something went wrong.
fi
exit $moduleresult
◇
```
Uses: moduleresult 64.

The run-script nlpp reads a “raw” naf from standard in and produces an annotated naf on standard out. It creates a temporary directory to store intermediate results from the modules and removes this directory afterwards.

"../bin/nlpp" 67b≡
```
#!/bin/bash
oldd=‘pwd‘
⟨ set variables for test/run script 67c ⟩
workdir=‘mktemp -d -t nlpp.XXXXXX‘
cd $workdir
cat >$workdir/$infile
⟨ function to run a module 64 ⟩
⟨ annotate 66a ⟩
if
  [ $moduleresult -eq 0 ]
then
  cat $outfile
fi
cd $oldd
rm -rf $workdir
exit $moduleresult
◇
```
Uses: moduleresult 64.

⟨ set variables for test/run script 67c ⟩ ≡
```
ROOT=/home/huygen/projecten/nlpp
source /home/huygen/projecten/nlpp/env/bin/progenv
BIND=$pipebin
infile=in.naf
outfile=out.naf
◇
```
Fragment referenced in 67ab.
Uses: BIND 64.

6.2 Logging

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

⟨ *variables of install-modules* 68a ⟩ ≡
```
LOGLEVEL=1
◇
```
Fragment referenced in 20a.

⟨ *logmess* 68b ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
◇
```
Fragment referenced in 8c, 9a, 68c.

6.3 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* 68c ⟩ ≡
```
SUCCES=0
cd $modulesdir
⟨ move module (68d $DIR ) 8a ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (68e Installed $DIR ) 68b ⟩
  ⟨ remove old module (68f $DIR ) 8b ⟩
else
  ⟨ re-instate old module (68g $DIR ) 8c ⟩
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 2 shows pathways to translate it into printable/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.

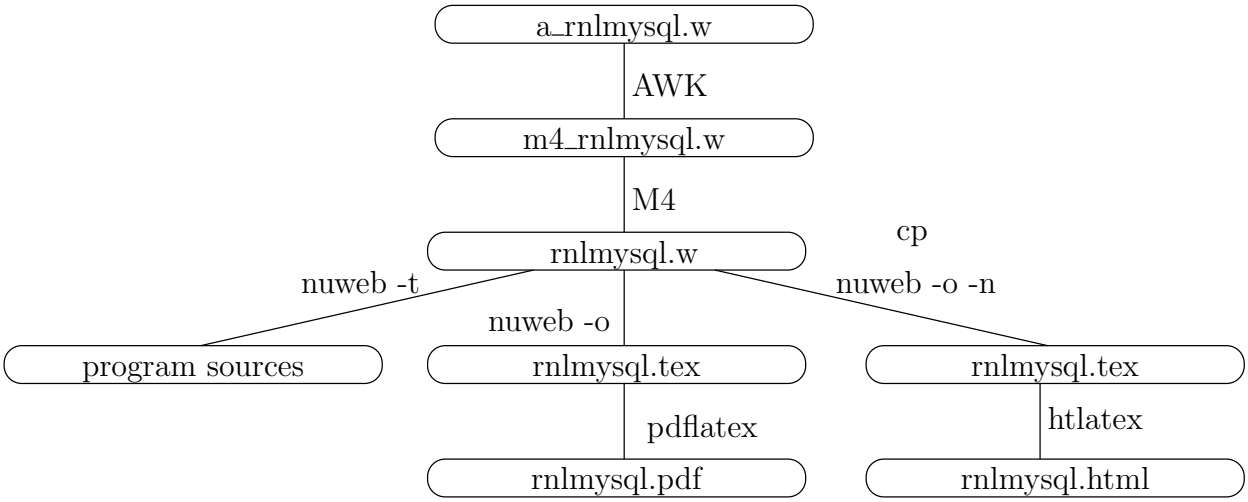

Figure 2: *Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.*

⟨ *parameters in Makefile* 70a ⟩ ≡
```
NUWEB=../env/bin/nuweb
```
◇
Fragment defined by 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Uses: `nuweb` 77b.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

`"Makefile"` 70b≡
⟨ *default target* 70c ⟩

⟨ *parameters in Makefile* 70a, ... ⟩

⟨ *impliciete make regels* 74a, ... ⟩
⟨ *expliciete make regels* 71d, ... ⟩
⟨ *make targets* 71a, ... ⟩
◇

The default target of make is `all`.

⟨ *default target* 70c ⟩ ≡
```
all : ⟨ all targets 71b ⟩
.PHONY : all

```
◇
Fragment referenced in 70b.
Defines: `all` 38b, `PHONY` 74b.

⟨ *make targets* 71a ⟩ ≡
```
clean:
        ⟨ clean up 12e, ... ⟩

◇
```
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.

The default is, to install nlpp.

⟨ *all targets* 71b ⟩ ≡
```
install◇
```
Fragment referenced in 70c.
Uses: `install` 81d.

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* 71c ⟩ ≡
```
.SUFFIXES: .pdf .w .tex .html .aux .log .php

◇
```
Fragment defined by 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 75a.

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.

⟨ *expliciete make regels* 71d ⟩ ≡
```
nuweb: $(NUWEB)

$(NUWEB): ../nuweb-1.58
        mkdir -p ../env/bin
        cd ../nuweb-1.58 && make nuweb
        cp ../nuweb-1.58/nuweb $(NUWEB)

◇
```
Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.
Uses: `nuweb` 77b.

⟨ *clean up* 72a ⟩ ≡
```
rm -rf ../nuweb-1.58
```
◇
Fragment defined by 12e, 13c, 28d, 72a.
Fragment referenced in 71a.
Uses: nuweb 77b.

⟨ *expliciete make regels* 72b ⟩ ≡
```
../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 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.
Uses: nuweb 77b.

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* 72c ⟩ ≡
```
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 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.
Uses: print 75a.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 72d ⟩ ≡
```
nlpp.w : m4_nlpp.w inst.m4
        m4 -P m4_nlpp.w > nlpp.w

```
◇
Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.

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* 73a ⟩ ≡

```
FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Defines: `FIGFILES` 73b.

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* 73b ⟩ ≡

```
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 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 75b, `PDF_FIG_NAMES` 75b, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 73a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 74a ⟩ ≡

```
%.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 74a, 78c.
Fragment referenced in 70b.
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* 74b ⟩ ≡

```
bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

.PHONY : bibfile
```

◇

Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.
Uses: `PHONY` 70c.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 75a ⟩ ≡
```
pdf : nlpp.pdf

print : nlpp.pdf
        lpr nlpp.pdf

view : nlpp.pdf
        evince nlpp.pdf

◇
```
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Defines: `pdf` 71c, 75b, `print` 14b, 16c, 19c, 27a, 35a, 57d, 72c, `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`.

⟨ *make targets* 75b ⟩ ≡
```
nlpp.pdf : nlpp.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

◇
```
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Uses: `pdf` 75a, `PDFT_NAMES` 73b, `PDF_FIG_NAMES` 73b.

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* 75c ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 6ab, 7ab, 12agh, 15d, 75c.
Fragment referenced in 81a.
Uses: `nuweb` 77b.

⟨ *parameters in Makefile* 75d ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Uses: `nuweb` 77b.

⟨ *expliciete make regels* 76a ⟩ ≡
```
$(W2PDF) : nlpp.w $(NUWEB)
        $(NUWEB) nlpp.w
```
◇
Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.

"../nuweb/bin/w2pdf" 76b≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20160530 at 1424h: Generated by nuweb from a_nlpp.w
NUWEB=../env/bin/nuweb
LATEXCOMPILER=pdflatex
```
⟨ *filenames in nuweb compile script* 76d ⟩
⟨ *compile nuweb* 76c ⟩

◇
Uses: `nuweb` 77b.

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* 76c ⟩ ≡
```
NUWEB=/home/huygen/projecten/nlpp/env/bin/nuweb
```
⟨ *run the processors until the aux file remains unchanged* 77c ⟩
⟨ *remove the copy of the aux file* 77a ⟩
◇
Fragment referenced in 76b.
Uses: `nuweb` 77b.

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* 76d ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
```
◇
Fragment referenced in 76b.
Defines: `auxfil` 77c, 79c, 80a, `indexfil` 77c, 79c, `nufil` 77b, 79c, 80b, `oldaux` 77ac, 79c, 80a, `oldindexfil` 77c, 79c, `texfil` 77b, 79c, 80b, `trunk` 77b, 79c, 80bc.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 77a ⟩ ≡

```
rm $oldaux
◇
```

Fragment referenced in 76c, 79b.
Uses: `oldaux` 76d, 79c.

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* 77b ⟩ ≡

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

Fragment referenced in 77c.
Defines: `bibtex` 80bc, `makeindex` 80bc, `nuweb` 7c, 66b, 70a, 71d, 72ab, 75cd, 76bc, 78a, 79a.
Uses: `nufil` 76d, 79c, `texfil` 76d, 79c, `trunk` 76d, 79c.

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* 77c ⟩ ≡

```
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 77b ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
◇
```

Fragment referenced in 76c.
Uses: `auxfil` 76d, 79c, `indexfil` 76d, `oldaux` 76d, 79c, `oldindexfil` 76d.

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* 78a ⟩ ≡
```
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 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Uses: `nuweb` 77b.

Make the directory:

⟨ *expliciete make regels* 78b ⟩ ≡
```
$(htmldir) :
        mkdir -p $(htmldir)
```
◇

Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.

The rule to copy files in it:

⟨ *impliciete make regels* 78c ⟩ ≡
```
$(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
```
◇

Fragment defined by 74a, 78c.
Fragment referenced in 70b.

Do the work:

⟨ *expliciete make regels* 78d ⟩ ≡
```
$(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
```
◇

Fragment defined by 71d, 72bcd, 74b, 76a, 78bd.
Fragment referenced in 70b.

Invoke:

⟨ *make targets* 78e ⟩ ≡
```
htm : $(htmldir) $(htmltarget)
```
◇

Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.

Create a script that performs the translation.

```
"w2html" 79a≡
        #!/bin/bash
        # w2html -- make a html file from a nuweb file
        # usage: w2html [filename]
        #  [filename]: Name of the nuweb source file.
        # 20160530 at 1424h: Generated by nuweb from a_nlpp.w
        echo "translate " $1 >w2html.log
        NUWEB=/home/huygen/projecten/nlpp/env/bin/nuweb
        ⟨ filenames in w2html 79c ⟩

        ⟨ perform the task of w2html 79b ⟩

        ◇
```

Uses: `nuweb` 77b.

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 79b ⟩ ≡
        ⟨ run the html processors until the aux file remains unchanged 80a ⟩
        ⟨ remove the copy of the aux file 77a ⟩
        ◇
```

Fragment referenced in 79a.

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 79c ⟩ ≡
        nufil=$1
        trunk=${1%%.*}
        texfil=${trunk}.tex
        auxfil=${trunk}.aux
        oldaux=old.${trunk}.aux
        indexfil=${trunk}.idx
        oldindexfil=old.${trunk}.idx
        ◇
```

Fragment referenced in 79a.
Defines: `auxfil` 76d, 77c, 80a, `nufil` 76d, 77b, 80b, `oldaux` 76d, 77ac, 80a, `texfil` 76d, 77b, 80b, `trunk` 76d, 77b, 80bc.
Uses: `indexfil` 76d, `oldindexfil` 76d.

⟨ *run the html processors until the aux file remains unchanged* 80a ⟩ ≡

```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 80b ⟩
done
⟨ run tex4ht 80c ⟩

◇
```

Fragment referenced in 79b.
Uses: `auxfil` 76d, 79c, `oldaux` 76d, 79c.

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

⟨ *run the html processors* 80b ⟩ ≡

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

Fragment referenced in 80a.
Uses: `bibtex` 77b, `makeindex` 77b, `nufil` 76d, 79c, `texfil` 76d, 79c, `trunk` 76d, 79c.

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* 80c ⟩ ≡

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

Fragment referenced in 80a.
Uses: `bibtex` 77b, `makeindex` 77b, `trunk` 76d, 79c.

A.7 Perform the installation

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* 80d ⟩ ≡

```
MKDIR = mkdir -p

◇
```

Fragment defined by 70a, 71c, 73ab, 75d, 78a, 80d.
Fragment referenced in 70b.
Defines: `MKDIR` 81a.

⟨ *make targets* 81a ⟩ ≡
```
DIRS = ⟨ directories to create 6a, ... ⟩

$(DIRS) :
        $(MKDIR) $@
```
◇
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Defines: `DIRS` 81c.
Uses: `MKDIR` 80d.

⟨ *make scripts executable* 81b ⟩ ≡
```
chmod -R 775  ../bin/*
chmod -R 775  ../env/bin/*
```
◇
Fragment defined by 24d, 34g, 81b.
Fragment referenced in 81c.

The target "sources" unpacks the nuweb file and creates the program scripts, i.e. the scripts that will apply modules on a NAF file and the script `install_modules` that installs the modules themselves and that creates the software environment the the modules need.

⟨ *make targets* 81c ⟩ ≡
```
sources : nlpp.w $(DIRS) $(NUWEB)
        $(NUWEB) nlpp.w
        ⟨ make scripts executable 24d, ... ⟩
```
◇
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Uses: `DIRS` 81a.

The "install" target performs the complete installation.

⟨ *make targets* 81d ⟩ ≡
```
install : sources
        ../bin/install-modules
```
◇
Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Defines: `install` 9c, 10b, 11a, 14c, 15b, 16bd, 17ab, 24dj, 25a, 29b, 31b, 41a, 56a, 57a, 58e, 59a, 61a, 63b, 71b, 82a.

A.8 Test whether it works

The targets testnl and testen perform the test-script (section **??**) to test the dutch resp. english pipeline.

⟨ *make targets* 82a ⟩ ≡

```
testnl : install test.nl.in.naf
        rm -rf ../test
        mkdir ../test
        cd ../test && ../bin/test nl

testen : install test.en.in.naf
        rm -rf ../test
        mkdir ../test
        cd ../test && ../bin/test en
```

◇

Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.
Defines: `testen` Never used, `testnl` Never used.
Uses: `install` 81d.

A.9 Restore paths after transplantation

When an existing installation has been transplanted to another location, many path indications have to be adapted to the new situation. The scripts that are generated by nuweb can be repaired by re-running nuweb. After that, configuration files of some modules must be modified.

⟨ *make targets* 82b ⟩ ≡

```
transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant
```

◇

Fragment defined by 71a, 75ab, 78e, 81acd, 82ab.
Fragment referenced in 70b.

In order to work as expected, the following script must be re-made after a transplantation.

"../env/bin/transplant" 82c≡

```
#!/bin/bash
LOGLEVEL=1
⟨ set variables that point to the directory-structure 7c, ... ⟩
⟨ set paths after transplantation 16c ⟩
⟨ re-install modules after the transplantation 31c ⟩
```

◇

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.

C Indexes

C.1 Filenames

C.2 Macro's

⟨ check whether program is present 24j ⟩ Referenced in 24e.
⟨ check/install the correct version of python 14b ⟩ Referenced in 14a.
⟨ clean up 12e, 13c, 28d, 72a ⟩ Referenced in 71a.
⟨ clone the heideltime wrapper 58b ⟩ Referenced in 58a.
⟨ compile nuweb 76c ⟩ Referenced in 76b.
⟨ compile the heideltime wrapper 58e ⟩ Referenced in 58a.
⟨ compile the nerc jar 52e ⟩ Referenced in 52d.
⟨ compile the topic-tool jar 41a ⟩ Referenced in 40a.
⟨ create a virtual environment for Python 15a ⟩ Referenced in 14a.
⟨ create javapython script 11e ⟩ Referenced in 20a.
⟨ default target 70c ⟩ Referenced in 70b.
⟨ directories to create 6ab, 7ab, 12agh, 15d, 75c ⟩ Referenced in 81a.
⟨ download svm models 54c ⟩ Referenced in 54a.
⟨ else conditional install 19a ⟩ Not referenced.
⟨ end conditional install 19b ⟩ Referenced in 12b, 20a, 21aj, 22aj, 23ahq, 24a.
⟨ expliciete make regels 71d, 72bcd, 74b, 76a, 78bd ⟩ Referenced in 70b.
⟨ filenames in nuweb compile script 76d ⟩ Referenced in 76b.
⟨ filenames in w2html 79c ⟩ Referenced in 79a.
⟨ find a spotlightserver or exit 35a ⟩ Referenced in 44a, 56b.
⟨ function to run a module 64 ⟩ Referenced in 67ab.
⟨ get a testfile or die 66b ⟩ Referenced in 67a.
⟨ get commandline-arguments 25b ⟩ Referenced in 33b.
⟨ get jvntextpro-2.0.jar 58f ⟩ Referenced in 58e.
⟨ get spotlight language parameters 35b ⟩ Not referenced.
⟨ get spotlight model ball 32d ⟩ Referenced in 32a.
⟨ get the mor time-out parameter 42a ⟩ Referenced in 41e.
⟨ get the nerc models 53b ⟩ Referenced in 52d.
⟨ get the path to the module-script 26c ⟩ Referenced in 26b.
⟨ get the snapshot 10a ⟩ Referenced in 20a.
⟨ impliciete make regels 74a, 78c ⟩ Referenced in 70b.
⟨ install ActivePython 14c ⟩ Referenced in 14b.
⟨ install Alpino 28b ⟩ Referenced in 20a.
⟨ install boost 31d ⟩ Referenced in 20a.
⟨ install coreference-base 52a ⟩ Referenced in 23a.
⟨ install CRFsuite 39a ⟩ Referenced in 20a.
⟨ install from github 9a ⟩ Referenced in 41d, 46g, 54a, 55d, 58b, 59e, 62c, 63a.
⟨ install from tarball 68c ⟩ Not referenced.
⟨ install Java 1.6 13e ⟩ Referenced in 20a.
⟨ install kafnafparserpy 16d ⟩ Referenced in 14a.
⟨ install libxml2 or libxslt 11a ⟩ Referenced in 11b.
⟨ install maven 13a ⟩ Referenced in 20a.
⟨ install perl 17bc, 18a ⟩ Referenced in 20a.
⟨ install python packages 17a, 61a ⟩ Referenced in 14a.
⟨ install sematree 26a ⟩ Referenced in 20a.
⟨ install shared libs 10b, 11b ⟩ Referenced in 20a.
⟨ install svm lib 54b ⟩ Referenced in 54a.
⟨ install SVMLight 38b ⟩ Referenced in 20a.
⟨ install the constituents parser 43a ⟩ Referenced in 21j.
⟨ install the dbpedia-ner module 62c ⟩ Referenced in 23q.
⟨ install the event-coreference module 61e ⟩ Referenced in 23h.
⟨ install the factuality module 51c ⟩ Referenced in 22j.
⟨ install the FBK-causalrel module 50c ⟩ Referenced in 22j.
⟨ install the FBK-temprel module 49b ⟩ Referenced in 22j.
⟨ install the FBK-time module 47c ⟩ Referenced in 22j.
⟨ install the heideltime module 58a ⟩ Referenced in 23h.
⟨ install the ims-wsd module 45a ⟩ Referenced in 22a.
⟨ install the jex resources and libraries 40b ⟩ Referenced in 40a.
⟨ install the lu2synset converter 55a ⟩ Referenced in 23h.

⟨ install the morphosyntactic parser 41d ⟩ Referenced in 21a.
⟨ install the NERC module 52d ⟩ Referenced in 21j.
⟨ install the ontotagger repository 57a ⟩ Referenced in 23a.
⟨ install the opinion-miner 63abc ⟩ Referenced in 24a.
⟨ install the pos tagger 42b ⟩ Referenced in 21a.
⟨ install the post-SRL module 61b ⟩ Referenced in 23q.
⟨ install the Spotlight server 32a, 33a ⟩ Referenced in 20a.
⟨ install the srl module 59e ⟩ Referenced in 23h.
⟨ install the srl-dutch-nominals module 46g ⟩ Referenced in 22a.
⟨ install the srl-server module 45d ⟩ Referenced in 22a.
⟨ install the ticcutils utility 30d ⟩ Referenced in 20a, 31c.
⟨ install the timbl utility 31a ⟩ Referenced in 20a, 31c.
⟨ install the tokenizer 39b ⟩ Referenced in 21a.
⟨ install the topic analyser 40a ⟩ Referenced in 21a.
⟨ install the treetagger utility 29abcd, 30abc ⟩ Referenced in 20a.
⟨ install the UKB module 44c ⟩ Not referenced.
⟨ install the wikify module 43g ⟩ Referenced in 21j.
⟨ install the WSD module 54a ⟩ Referenced in 23a.
⟨ install the NED-reranker module 43d ⟩ Referenced in 21j.
⟨ install the NED module 55d ⟩ Referenced in 21j.
⟨ install VUA-pylib 38a ⟩ Referenced in 20a.
⟨ logmess 68b ⟩ Referenced in 8c, 9a, 68c.
⟨ make scripts executable 24d, 34g, 81b ⟩ Referenced in 81c.
⟨ make targets 71a, 75ab, 78e, 81acd, 82ab ⟩ Referenced in 70b.
⟨ move module 8a ⟩ Referenced in 9a, 68c.
⟨ parameters in Makefile 70a, 71c, 73ab, 75d, 78a, 80d ⟩ Referenced in 70b.
⟨ perform the task of w2html 79b ⟩ Referenced in 79a.
⟨ put Antske's material in the heideltime wrapper 58d ⟩ Referenced in 58a.
⟨ put spotlight jar in the Maven repository 56a ⟩ Referenced in 55d.
⟨ re-install modules after the transplantation 31c ⟩ Referenced in 82c.
⟨ re-instate old module 8c ⟩ Referenced in 9a, 68c.
⟨ read the list of installed modules 18b ⟩ Referenced in 20a.
⟨ remove installed-variable 19c ⟩ Referenced in 13c.
⟨ remove old module 8b ⟩ Referenced in 9a, 68c.
⟨ remove outdated heideltime jars 59b ⟩ Referenced in 59a.
⟨ remove the copy of the aux file 77a ⟩ Referenced in 76c, 79b.
⟨ repair causalrel's run.sh.hadoop 50e ⟩ Not referenced.
⟨ repair FBK-*rel's run.sh.hadoop 49d ⟩ Referenced in 49b, 50c.
⟨ run in subshell when naflang is not known 27b ⟩ Referenced in 26b.
⟨ run only if language is English or Dutch 28a ⟩ Referenced in 26b.
⟨ run tex4ht 80c ⟩ Referenced in 80a.
⟨ run the html processors 80b ⟩ Referenced in 80a.
⟨ run the html processors until the aux file remains unchanged 80a ⟩ Referenced in 79b.
⟨ run the processors until the aux file remains unchanged 77c ⟩ Referenced in 76c.
⟨ run the three processors 77b ⟩ Referenced in 77c.
⟨ select language-dependent features 53a ⟩ Not referenced.
⟨ set alpinohome 28c ⟩ Referenced in 41e.
⟨ set default arguments for Spotlight 34a ⟩ Referenced in 33b.
⟨ set paths after transplantation 16c ⟩ Referenced in 82c.
⟨ set up java 12bf ⟩ Referenced in 20a.
⟨ set up Java 1.6 13f ⟩ Referenced in 45b.
⟨ set up python 14a ⟩ Referenced in 20a.
⟨ set variables for test/run script 67c ⟩ Referenced in 67ab.
⟨ set variables that point to the directory-structure 7cd, 9f, 13b ⟩ Referenced in 7e, 20a, 82c.
⟨ start EHU SRL server if it isn't running 46e ⟩ Referenced in 45e.
⟨ start of module-script 26b ⟩ Referenced in 39c, 41be, 42c, 43be, 44ad, 45be, 46ac, 47a, 48a, 50a, 51ad, 52b, 53ce, 54d, 55b, 56b, 57bdf, 59c, 60a, 61c, 62ad, 63d.
⟨ start the Spotlight server on localhost 37ab ⟩ Referenced in 34b, 36a.

⟨ stop EHU SRL server 46f ⟩ Referenced in 46a.
⟨ stop on error 49a ⟩ Referenced in 48a.
⟨ test whether spotlighthost runs 36e ⟩ Referenced in 36a.
⟨ test whether virtualenv is present on the host 15b ⟩ Referenced in 15a.
⟨ try to obtain a running spotlightserver 36a ⟩ Not referenced.
⟨ unpack ticcutils or timbl 31b ⟩ Referenced in 30d, 31a.
⟨ update pip 16b ⟩ Referenced in 14a.
⟨ variables of install-modules 68a ⟩ Referenced in 20a.
⟨ wait until the spotlight server is up or faulty 37c ⟩ Referenced in 37b.

C.3 Variables