# Bilingual NLP pipeline

**Paul Huygen <paul.huygen@huygen.nl>**

**10th May 2017**
**13:59 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.   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 | 5.6.23 | 0fd3... | heideltime | nl |
| Onto-tagger | https://github.com/cltl/OntoTagger.git | 5.6.22 | 9ea0... | onto | nl |
| Semantic Role labeling nl | https://github.com/newsreader/vua-srl-nl.git | 5.6.24 | 675d... | srl | nl |
| Semantic Role labeling en | snapshot | 5.6.10 | ... | eSRL | en |
| Nominal Event ann. | https://github.com/cltl/OntoTagger.git | 5.6.22 | 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 | 5.6.22 | 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 | 5.6.29 | 40a7... | opinimin | en/nl |
| Event-coref | snapshot | 5.6.26 | ... | 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.*

### 1.2    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.

### 1.3    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
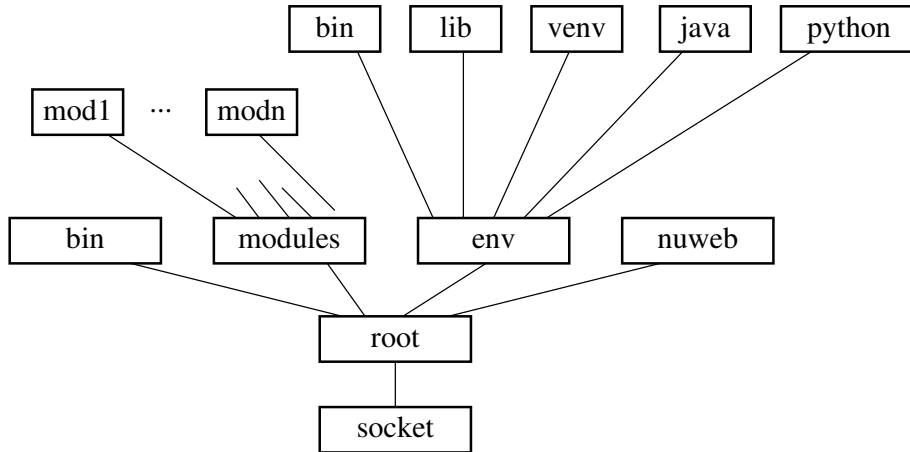


Figure 1: *Directory-structure of the pipeline (see text).*

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* 6 ⟩ ≡
        `../modules` ⋄
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

⟨ *directories to create* 7a ⟩ ≡
```
        ../bin ../env/bin ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

⟨ *directories to create* 7b ⟩ ≡
```
        ../env/lib ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

⟨ *directories to create* 7c ⟩ ≡
```
        ../env/etc ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

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* 7d ⟩ ≡
```
        if
          [ "$piperoot" == "" ]
        then
          export piperoot=/home/paul/projecten/pipelines/nlpp
        fi
        export pipesocket=${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 7de, 10f, 14h.
Fragment referenced in 8a, 22a, 90.
Uses: `nuweb` 84d.

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

⟨ *set variables that point to the directory-structure* 7e ⟩ ≡
```
        export PATH=$envbindir:$PATH
        ◇
```
Fragment defined by 7de, 10f, 14h.
Fragment referenced in 8a, 22a, 90.
Defines: `PATH` 14ah, 15d, 19f, 56a.

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

```
"../env/bin/progenv" 8a≡
      #!/bin/bash
      ⟨ set variables that point to the directory-structure 7d, ... ⟩
      export progenvset=0
      ◇
```
File defined by 8a, 13b, 48a.

## 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 8b ⟩ ≡
      if
       [ -e @1 ]
      then
         mv @1 old.@1
      fi
      ◇
```
Fragment referenced in 10a, 76a.

```
⟨ remove old module 8c ⟩ ≡
      rm -rf old.@1
      ◇
```
Fragment referenced in 10a, 76a.

⟨ *re-instate old module* 9a ⟩ ≡

```
mv old.@1 @1
MESS="Replaced previous version of @1"
```
⟨ *logmess* (9b $MESS ) 75d ⟩

◇

Fragment referenced in 10a, 76a.

## 2.3   Download materials

This installer needs to download a lot from different sources:

- Most of the NLP-modules will be built up from their sources in Github. The sources must be cloned.
- Many modules need external resources, e.g. the Alpino tagger. Often these utilities must be downloaded from a location specified by the supplier.
- Many modules use extra resources like model-data, that must be obtained separately.
- Some of the resources are not publicly available. They must be obtained from a pass-word protected URL.
-

Usually downloads are slow, and the duration is only little determined by the resources in the installing computer, but by the network and the performance of the systems from which we download. Therefore, we may speed up by first downloading things, if possible in parallel processes.

We put the following the beginning of the install-script:

⟨ *download everything* 9c ⟩ ≡

```
⟨ download stuff 11a, ... ⟩
echo Waiting for downloads to complete ...
wait
echo Download completed
```
◇

Fragment referenced in 22a.

## 2.4   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* 10a ⟩ ≡

```
      cd $modulesdir
```
      ⟨ *move module* (10b $DIRN ) 8b ⟩
```
      git clone $GITU
      if
        [ $? -gt 0 ]
      then
```
        ⟨ *logmess* (10c Cannot install current $MODNAM version ) 75d ⟩
        ⟨ *re-instate old module* (10d $DIRN ) 9a ⟩
```
      else
```
        ⟨ *remove old module* (10e $DIRN ) 8c ⟩
```
        cd $modulesdir/$DIRN
        git checkout $GITC
      fi
```

◇

Fragment referenced in 43d, 50f, 55c, 58e, 60c, 63d, 65a, 67d, 69c.

## 2.5   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* 10f ⟩ ≡

```
      if
        [ ! $snapshotsocket ]
      then
        export snapshotsocket=/home/paul/projecten/pipelines
      fi
      if
        [ ! $snapshotdirectory ]
      then
        export snapshotdirectory=t_nlpp_resources
      fi
```
◇

Fragment defined by 7de, 10f, 14h.
Fragment referenced in 8a, 22a, 90.

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* 10g ⟩ ≡

```
      if
        [ ! -e /home/paul/projecten/pipelines/nrkey ]
      then
        echo "No key to connect to snapshot!"
        exit 1
      fi
```
◇

Fragment defined by 10g, 26e.
Fragment referenced in 22a.

Update the local snapshot repository.

⟨ *download stuff* 11a ⟩ ≡

```
cd $snapshotsocket
mkdir -p $snapshotdirectory
( rsync -e "ssh -i /home/paul/projecten/pipelines/nrkey" -
rLt newsreader@kyoto.let.vu.nl:t_nlpp_resources . ) &
```
◇

Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

## 2.6 Download other materials

Apart from the material that we obtain from the snapshot, we need to download resources from different places in the Internet. Downloading can take much time. While working on this installer, we do not want to repeat downloading every time that we run it e.g. to test something. Therefore, we download everything in the snapshot-directory, and check whether it is already there before we start downloading.

⟨ *need to wget* 11b ⟩ ≡

```
if
  [ ! -e $snapshotsocket/$snapshotdirectory/@1 ]
then
  cd $snapshotsocket/$snapshotdirectory
  ( wget @2 ) &
fi
```
◇

Fragment referenced in 11c, 14c, 19b, 34b, 40e, 42b, 60d.

## 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`.

Get autoconf:

⟨ *download stuff* 11c ⟩ ≡

```
⟨ need to wget (11d autoconf-2.69.tar.gz,11e http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz ) 11b ⟩
```
◇

Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

Install autoconf:

⟨ *install shared libs* 12a ⟩ ≡

```
        autoconfdir=`mktemp -d -t autoconf.XXXXXX`
        cd $autoconfdir
        tar -xzf $snapshotsocket/$snapshotdirectory/autoconf-2.69.tar.gz
        cd autoconf-2.69
        ./configure --prefix=$envdir
        make
        make install
        cd $piperoot
        rm -rf $autoconfdir
        ◇
```
Fragment defined by 12ac.
Fragment referenced in 22a.
Uses: `install` 89a.

## 3.2   libxml2 and libxslt

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

⟨ *install libxml2 or libxslt* 12b ⟩ ≡

```
        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 12c.
Uses: `install` 89a.

⟨ *install shared libs* 12c ⟩ ≡

```
        ⟨ install libxml2 or libxslt (12d git://git.gnome.org/libxml2 ) 12b ⟩
        ⟨ install libxml2 or libxslt (12e git://git.gnome.org/libxslt ) 12b ⟩
        ◇
```
Fragment defined by 12ac.
Fragment referenced in 22a.

## 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* 13a ⟩ ≡
```
echo '#!/bin/bash' > /home/paul/projecten/pipelines/nlpp/env/bin/javapython
```
◇

Fragment referenced in 22a.

Cause the module scripts to read the javapython script.

"../env/bin/progenv" 13b≡
```
source $envbindir/javapython
```
◇

File defined by 8a, 13b, 48a.

## 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* 13c ⟩ ≡
```
../env/java ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

⟨ *set up java* 13d ⟩ ≡
```
⟨ begin conditional install (13e java_installed ) 20c ⟩
  cd $envdir/java
  tar -xzf $snapshotsocket/$snapshotdirectory/server-jre-7u72-linux-x64.tar.gz
⟨ end conditional install (13f java_installed ) 21b ⟩
```
◇

Fragment defined by 13d, 14a.
Fragment referenced in 22a.

Remove the java-ball when cleaning up:

⟨ *clean up* 13g ⟩ ≡
```
rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz
```
◇

Fragment defined by 13g, 15a, 31b, 79c.
Fragment referenced in 78d.

Set variables for Java.

⟨ *set up java* 14a ⟩ ≡

```
echo 'export JAVA_HOME=$envdir/java/jdk1.7.0_72' >> /home/paul/projecten/pipelines/nlpp/env/bin/javap
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /home/paul/projecten/pipelines/nlpp/env/bin/javapython
export JAVA_HOME=$envdir/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 13d, 14a.
Fragment referenced in 22a.
Uses: `PATH` 7e.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 14b ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

## 4.2   Maven

Some Java-based modules can best be compiled with Maven. So download and install Maven:

⟨ *download stuff* 14c ⟩ ≡

```
⟨ need to wget (14d apache-maven-3.0.5-bin.tar.gz,14e http://apache.rediris.es/maven/maven-3/3.0.5/binarie
◇
```
Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

⟨ *directories to create* 14f ⟩ ≡
```
../env/apache-maven-3.0.5 ◇
```
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

⟨ *install maven* 14g ⟩ ≡
```
cd $envdir
tar -xzf /home/paul/projecten/pipelines/t_nlpp_resources/apache-maven-3.0.5-
bin.tar.gz
◇
```
Fragment referenced in 22a.

⟨ *set variables that point to the directory-structure* 14h ⟩ ≡
```
export MAVEN_HOME=$envdir/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```
Fragment defined by 7de, 10f, 14h.
Fragment referenced in 8a, 22a, 90.
Uses: `PATH` 7e.

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

⟨ *clean up* 15a ⟩ ≡
```
rm -rf ../env/apache-maven-3.0.5
```
⟨ *remove installed-variable* (15b `maven_installed` ) 21c ⟩
◇

Fragment defined by 13g, 15a, 31b, 79c.
Fragment referenced in 78d.

## 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* 15c ⟩ ≡
```
cd $envdir/java
$snapshotsocket/t_nlpp_resources/jre-6u45-linux-x64.bin
```
◇

Fragment referenced in 22a.

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

⟨ *set up Java 1.6* 15d ⟩ ≡
```
export JAVA_HOME=$envdir/java/jre1.6.0_45
export PATH=$JAVA_HOME/bin:$PATH
```
◇

Fragment referenced in 47b.
Uses: `PATH` 7e.

## 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* 15e ⟩ ≡
   ⟨ *check/install the correct version of python* 16a ⟩
   ⟨ *create a virtual environment for Python* 16c ⟩
   ⟨ *activate the python environment* 17b, . . . ⟩
   ⟨ *update pip* 18a ⟩
   ⟨ *install python packages* 19a, . . . ⟩
   ⟨ *install kafnafparserpy* 18c ⟩
   ◇

Fragment referenced in 22a.

⟨ *check/install the correct version of python* 16a ⟩ ≡
```
      pythonok=`python --
      version 2>&1 | gawk '{if(match($2, "2.7")) print "yes" ; else print "no" }'`
      if
        [ "$pythonok" == "no" ]
      then
        ⟨ install ActivePython 16b ⟩
      fi
      ◇
```
Fragment referenced in 15e.
Defines: pythonok Never used.
Uses: print 82b.

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* 16b ⟩ ≡
```
      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 16a.
Uses: install 89a, virtualenv 17a.

### 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* 16c ⟩ ≡
```
      ⟨ test whether virtualenv is present on the host 17a ⟩
      cd $envdir
      virtualenv venv
      ◇
```
Fragment referenced in 15e.
Uses: virtualenv 17a.

⟨ *test whether virtualenv is present on the host* 17a ⟩ ≡
```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
```
◇

Fragment referenced in 16c.
Defines: `virtualenv` 16bc.
Uses: `install` 89a.

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

⟨ *activate the python environment* 17b ⟩ ≡
```
source $envdir/venv/bin/activate
echo 'source $en-
vdir/venv/bin/activate' >> /home/paul/projecten/pipelines/nlpp/env/bin/javapython
```
◇

Fragment defined by 17bde.
Fragment referenced in 15e, 22a.
Defines: `activate` 18b.

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

⟨ *directories to create* 17c ⟩ ≡
```
../env/python
```
◇
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.

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

⟨ *activate the python environment* 17d ⟩ ≡
```
echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /home/paul/projecten/pipelines/nlpp/env/bin/javapytho
export PYTHONPATH=$envdir/python:$PYTHONPATH
```
◇

Fragment defined by 17bde.
Fragment referenced in 15e, 22a.
Defines: `PYTHONPATH` Never used.

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

⟨ *activate the python environment* 17e ⟩ ≡
```
echo ex-
port 'LD_LIBRARY_PATH=$envlibdir:$LD_LIBRARY_PATH' >> /home/paul/projecten/pipelines/nlpp/env/bin/jav
export LD_LIBRARY_PATH=$envdir/python:$LD_LIBRARY_PATH
```
◇

Fragment defined by 17bde.
Fragment referenced in 15e, 22a.
Defines: `LD_LIBRARY_PATH` 46d, 56a.

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

⟨ *update pip* 18a ⟩ ≡
```
pip install --upgrade pip
```
      ◇

Fragment referenced in 15e.
Uses: `install` 89a.

### 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* 18b ⟩ ≡
```
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 90.
Uses: `activate` 17b, `print` 82b.

### 4.4.3   KafNafParserPy

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

⟨ *install kafnafparserpy* 18c ⟩ ≡
```
pip install KafNafParserPy
```
      ◇

Fragment referenced in 15e.
Uses: `install` 89a.

### 4.4.4   Python packages

Install python packages:

**lxml:**
**pyyaml:** for coreference-graph
**pynaf:**
**requests:** for networkx

**networkx:** for corefbase.

⟨ *install python packages* 19a ⟩ ≡
```
      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 19a, 66a.
Fragment referenced in 15e.
Defines: lxml Never used, networkx Never used, pyyaml Never used.
Uses: install 89a.

## 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).

⟨ *download stuff* 19b ⟩ ≡

      ⟨ *need to wget* (19c perl-5.22.1.tar.gz,19d http://www.cpan.org/src/5.0/perl-5.22.1.tar.gz ) 11b ⟩

    ◇
Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

⟨ *install perl* 19e ⟩ ≡
```
      tempdir='mktemp -d -t perl.XXXXXX'
      cd $tempdir
      tar -xzf $snapshotsocket/$snapshotdirectory/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 19ef, 20a.
Fragment referenced in 22a.
Uses: install 89a.

Make sure that modules use the correct Perl

⟨ *install perl* 19f ⟩ ≡
```
      echo 'export PERL_HOME=$envdir/perl' >> /home/paul/projecten/pipelines/nlpp/env/bin/javapython
      echo 'export PATH=$PERL_HOME/bin:$PATH' >> /home/paul/projecten/pipelines/nlpp/env/bin/javapython
      export PERL_HOME=$envdir/perl
      export PATH=$PERL_HOME/bin:$PATH
      ◇
```
Fragment defined by 19ef, 20a.
Fragment referenced in 22a.
Uses: PATH 7e.

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* 20a ⟩ ≡
```
    cd $envdir/perl/lib
    tar -xzf $snapshotsocket/t_nlpp_resources/20160520_nlpp_perllib.tgz
    ◇
```
Fragment defined by 19ef, 20a.
Fragment referenced in 22a.

# 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/paul/projecten/pipelines/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* 20b ⟩ ≡
```
    if
      [ -e /home/paul/projecten/pipelines/nlpp/installed_modules ]
    then
      source /home/paul/projecten/pipelines/nlpp/installed_modules
    fi
    ◇
```
Fragment referenced in 22a.

⟨ *begin conditional install* 20c ⟩ ≡
```
    if
      [ ! $@1 ]
    then
    ◇
```
Fragment referenced in 13d, 22a, 23aj, 24aj, 25ahq, 26a.

⟨ *else conditional install* 21a ⟩ ≡
```
      else
      ◇
```
Fragment never referenced.

⟨ *end conditional install* 21b ⟩ ≡
```
        echo "export @1=0" >> /home/paul/projecten/pipelines/nlpp/installed_modules
      fi
      ◇
```
Fragment referenced in 13d, 22a, 23aj, 24aj, 25ahq, 26a.

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

⟨ *remove installed-variable* 21c ⟩ ≡
```
      cd $piperoot
      mv /home/paul/projecten/pipelines/nlpp/installed_modules old.modulelist
      cat old.modulelist | gawk '/@1/ {next}; {print}' >/home/paul/projecten/pipelines/nlpp/installed_modul
      ◇
```
Fragment referenced in 15a.
Uses: print 82b.

## 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" 22a≡
    #!/bin/bash
    echo Set up environment
    ⟨ set variables that point to the directory-structure 7d, … ⟩
    ⟨ read the list of installed modules 20b ⟩
    ⟨ check this first 10g, … ⟩
    ⟨ download everything 9c ⟩
    ⟨ variables of install-modules 75c ⟩
    ⟨ begin conditional install (22b shared_libs ) 20c ⟩
        ⟨ install shared libs 12a, … ⟩
    ⟨ end conditional install (22c shared_libs ) 21b ⟩
    ⟨ create javapython script 13a ⟩
    echo ... Java
    ⟨ set up java 13d, … ⟩
    ⟨ begin conditional install (22d maven_installed ) 20c ⟩
        ⟨ install maven 14g ⟩
    ⟨ end conditional install (22e maven_installed ) 21b ⟩
    ⟨ begin conditional install (22f java16_installed ) 20c ⟩
        ⟨ install Java 1.6 15c ⟩
    ⟨ end conditional install (22g java16_installed ) 21b ⟩


    echo ... Python
    if
        [ $python_installed ]
    then
        ⟨ activate the python environment 17b, … ⟩
    fi
    ⟨ begin conditional install (22h python_installed ) 20c ⟩
        ⟨ set up python 15e ⟩
    ⟨ end conditional install (22i python_installed ) 21b ⟩
    ⟨ begin conditional install (22j perl_installed ) 20c ⟩
        ⟨ install perl 19e, … ⟩
    ⟨ end conditional install (22k perl_installed ) 21b ⟩


    ⟨ begin conditional install (22l sematree_installed ) 20c ⟩
        ⟨ install sematree 28a ⟩
    ⟨ end conditional install (22m sematree_installed ) 21b ⟩
    echo ... Alpino
    ⟨ begin conditional install (22n alpino_installed ) 20c ⟩
        ⟨ install Alpino 30c ⟩
    ⟨ end conditional install (22o alpino_installed ) 21b ⟩
    echo ... Spotlight
    ⟨ begin conditional install (22p spotlight_installed ) 20c ⟩
        ⟨ install the Spotlight server 35a, … ⟩
    ⟨ end conditional install (22q spotlight_installed ) 21b ⟩
    echo ... Treetagger
    ⟨ begin conditional install (22r treetagger_installed ) 20c ⟩
        ⟨ install the treetagger utility 31c, … ⟩
    ⟨ end conditional install (22s treetagger_installed ) 21b ⟩
    echo ... Ticcutils and Timbl
    ⟨ begin conditional install (22t ticctimbl_installed ) 20c ⟩
        ⟨ install the ticcutils utility 33a ⟩
        ⟨ install the timbl utility 33b ⟩
    ⟨ end conditional install (22u ticctimbl_installed ) 21b ⟩
    echo ... Boost
    ⟨ begin conditional install (22v boost_installed ) 20c ⟩
        ⟨ install boost 34a ⟩
    ⟨ end conditional install (22w boost_installed ) 21b ⟩
    echo ... VUA-pylib, SVMlight, CRFsuite
    ⟨ begin conditional install (22x miscutils_installed ) 20c ⟩
        ⟨ install VUA-pylib 40d ⟩
        ⟨ install SVMLight 41a ⟩
    ⟨ end conditional install (22y miscutils_installed ) 21b ⟩
    ◇
```
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

Next, install the modules:

```
"../bin/install-modules" 23a≡
        echo Install modules
        ⟨ begin conditional install (23b tokenizer_installed ) 20c ⟩
          echo ... Tokenizer
          ⟨ install the tokenizer 41b ⟩
        ⟨ end conditional install (23c tokenizer_installed ) 21b ⟩
        ⟨ begin conditional install (23d topic_installed ) 20c ⟩
          echo ... Topic detector
          ⟨ install the topic analyser 42a ⟩
        ⟨ end conditional install (23e topic_installed ) 21b ⟩
        ⟨ begin conditional install (23f morpar_installed ) 20c ⟩
          echo ... Morphosyntactic parser
          ⟨ install the morphosyntactic parser 43d ⟩
        ⟨ end conditional install (23g morpar_installed ) 21b ⟩
        ⟨ begin conditional install (23h pos_installed ) 20c ⟩
          echo "... Pos tagger (for english docs)"
          ⟨ install the pos tagger 44b ⟩
        ⟨ end conditional install (23i pos_installed ) 21b ⟩
        ◇
```
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

```
"../bin/install-modules" 23j≡
        ⟨ begin conditional install (23k constparse_installed ) 20c ⟩
          echo "... Constituent parser (for english docs)"
          ⟨ install the constituents parser 45a ⟩
        ⟨ end conditional install (23l constparse_installed ) 21b ⟩
        ⟨ begin conditional install (23m nerc_installed ) 20c ⟩
          echo ... NERC
          ⟨ install the NERC module 57a ⟩
        ⟨ end conditional install (23n nerc_installed ) 21b ⟩
        ⟨ begin conditional install (23o ned_installed ) 20c ⟩
          echo ... NED
          ⟨ install the NED module 60c ⟩
        ⟨ end conditional install (23p ned_installed ) 21b ⟩
        ⟨ begin conditional install (23q der_installed ) 20c ⟩
          echo ... Dark-entity relinker
          ⟨ install the de-link module 61d ⟩
        ⟨ end conditional install (23r der_installed ) 21b ⟩
        ⟨ begin conditional install (23s nedrer_installed ) 20c ⟩
          echo ...NED reranker
          ⟨ install the NED-reranker module 45d ⟩
        ⟨ end conditional install (23t nedrer_installed ) 21b ⟩
        ⟨ begin conditional install (23u wikify_installed ) 20c ⟩
          echo ...WIKIfy module
          ⟨ install the wikify module 45g ⟩
        ⟨ end conditional install (23v wikify_installed ) 21b ⟩
        ◇
```
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

"../bin/install-modules" 24a≡
        ⟨ *begin conditional install* (24b `UKB_installed` ) 20c ⟩
          echo ... UKB module
         cd $modulesdir
         tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-ukb.v30.tgz
        ⟨ *end conditional install* (24c `UKB_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24d `ims_wsd_installed` ) 20c ⟩
          echo ...ims-wsd module
          ⟨ *install the ims-wsd module* 47a ⟩
        ⟨ *end conditional install* (24e `ims_wsd_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24f `srl_server_installed` ) 20c ⟩
          echo ...srl-server module
          ⟨ *install the srl-server module* 49c ⟩
        ⟨ *end conditional install* (24g `srl_server_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24h `srl_dutch_nominals_installed` ) 20c ⟩
          echo ...srl-dutch-nominal module
          ⟨ *install the srl-dutch-nominals module* 50f ⟩
        ⟨ *end conditional install* (24i `srl_dutch_nominals_installed` ) 21b ⟩
        ◇
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

"../bin/install-modules" 24j≡
        ⟨ *begin conditional install* (24k `FBK_time_installed` ) 20c ⟩
          echo ... FBK-time module
          ⟨ *install the FBK-time module* 51c ⟩
        ⟨ *end conditional install* (24l `FBK_time_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24m `FBK_temprel_installed` ) 20c ⟩
          echo ... FBK-temprel module
          ⟨ *install the FBK-temprel module* 53b ⟩
        ⟨ *end conditional install* (24n `FBK_temprel_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24o `FBK_causalrel_installed` ) 20c ⟩
          echo ... FBK-causalrel module
          ⟨ *install the FBK-causalrel module* 54c ⟩
        ⟨ *end conditional install* (24p `FBK_causalrel_installed` ) 21b ⟩
        ⟨ *begin conditional install* (24q `factuality_installed` ) 20c ⟩
          echo ... factuality module
          ⟨ *install the factuality modules* 55c ⟩
        ⟨ *end conditional install* (24r `factuality_installed` ) 21b ⟩
        ◇
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

"../bin/install-modules" 25a≡
  ⟨ *begin conditional install* (25b `corefb_installed` ) 20c ⟩
   echo ... Coreference base
   ⟨ *install coreference-base* 56c ⟩
  ⟨ *end conditional install* (25c `corefb_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25d `wsd_installed` ) 20c ⟩
   echo ... WSD
   ⟨ *install the WSD module* 58e ⟩
  ⟨ *end conditional install* (25e `wsd_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25f `ontojar_installed` ) 20c ⟩
   echo ... Ontotagger
   ⟨ *install the ontotagger repository* 62c ⟩
  ⟨ *end conditional install* (25g `ontojar_installed` ) 21b ⟩
  ◇
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

"../bin/install-modules" 25h≡
  ⟨ *begin conditional install* (25i `heidel_installed` ) 20c ⟩
   echo ... Heideltime
   ⟨ *install the heideltime module* 63c ⟩
  ⟨ *end conditional install* (25j `heidel_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25k `SRL_installed` ) 20c ⟩
   echo ... SRL
   ⟨ *install the srl module* 65a ⟩
  ⟨ *end conditional install* (25l `SRL_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25m `eventcoref_installed` ) 20c ⟩
   echo ... Event-coreference
   ⟨ *install the event-coreference module* 67a ⟩
  ⟨ *end conditional install* (25n `eventcoref_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25o `lu2synset_installed` ) 20c ⟩
   echo ... lu2synset
   ⟨ *install the lu2synset converter* 59e ⟩
  ⟨ *end conditional install* (25p `lu2synset_installed` ) 21b ⟩
  ◇
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

"../bin/install-modules" 25q≡
  ⟨ *begin conditional install* (25r `dbpner_installed` ) 20c ⟩
   echo ... dbpedia-ner
   ⟨ *install the dbpedia-ner module* 67d ⟩
  ⟨ *end conditional install* (25s `dbpner_installed` ) 21b ⟩
  ⟨ *begin conditional install* (25t `post_SRL_installed` ) 20c ⟩
   echo ... post-SRL
   ⟨ *install the post-SRL module* 66b ⟩
  ⟨ *end conditional install* (25u `post_SRL_installed` ) 21b ⟩
  ◇
File defined by 22a, 23aj, 24aj, 25ahq, 26a.

```
"../bin/install-modules" 26a≡
        ⟨ begin conditional install (26b opimin_installed ) 20c ⟩
            echo ... opinion-miner
            ⟨ install the opinion-miner 69c, ... ⟩
        ⟨ end conditional install (26c opimin_installed ) 21b ⟩

        echo Final
        ◇
```
File defined by 22a, 23aj, 24aj, 25ahq, 26a.


```
⟨ make scripts executable 26d ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```
Fragment defined by 26d, 37a, 88c.
Fragment referenced in 88d.
Uses: install 89a.


## 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 26e ⟩ ≡
        ⟨ check whether program is present (26f git ) 26j ⟩
        ⟨ check whether program is present (26g tar ) 26j ⟩
        ⟨ check whether program is present (26h unzip ) 26j ⟩
        ⟨ check whether program is present (26i tcsh ) 26j ⟩
        ⟨ check whether mercurial is present 27a ⟩
        ◇
```
Fragment defined by 10g, 26e.
Fragment referenced in 22a.


```
⟨ check whether program is present 26j ⟩ ≡
        which @1
        if
          [ $? -ne 0 ]
        then
          echo Please install @1.
          exit 1
        fi
        ◇
```
Fragment referenced in 26e.
Uses: install 89a.

⟨ *check whether mercurial is present* 27a ⟩ ≡

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

Fragment referenced in 26e.
Defines: `hg` Never used.
Uses: `install` 89a.

## 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* 27b ⟩ ≡

```
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 36a.
Uses: `naflang` 73.

### 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, e.g. when multiple compute nodes use the same file-system, we need a directory on the local filesystem of the node itself. In that case you can set variable `semaworkdir` to override the default directory.

⟨ *install sematree* 28a ⟩ ≡
```
cat $snapshotsocket/t_nlpp_resources/sematree \
| sed "s|workdir|semaworkdir|g" \
| sed "s|/var/run|/home/paul/projecten/pipelines/nlpp/env/etc/sematree|g" \
> $envbindir/sematree
chmod 775 $envbindir/sematree
◇
```
Fragment referenced in 22a.

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* 28b ⟩ ≡
```
#!/bin/bash
⟨ get the path to the module-script 29a ⟩
source /home/paul/projecten/pipelines/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 30a ⟩
⟨ run only if language is English or Dutch 30b ⟩
◇
```
Fragment referenced in 41c, 43be, 44c, 45be, 46ad, 47b, 50c, 51a, 52a, 54a, 55ad, 56d, 58ac, 59c, 60a, 61b, 62adf, 63a, 64d, 65b, 66c, 67b, 68a, 69a, 71a.

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* 29a ⟩ ≡
```
      scriptdir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
      scriptname=${0##*/}
      scriptpath=$scriptdir/$scriptname
      ◇
```
Fragment referenced in 28b.
Defines: scriptpath 30a.

### 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"` 29b≡
```
      #!/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 82b.

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* 30a ⟩ ≡
```
if
  [ -z "${naflang+x}" ]
then
  naffile=`mktemp -t naf.XXXXXX`
  cat >$naffile
  naflang=`cat $naffile | python $envbindir/langdetect.py`
  export naflang
  cat $naffile | $scriptpath
  result=$?
  rm $naffile
  exit $result
fi
◇
```
Fragment referenced in 28b.
Uses: `naflang` 73, `scriptpath` 29a.

⟨ *run only if language is English or Dutch* 30b ⟩ ≡
```
if
  [ ! "$naflang" == "nl" ] && [ ! "$naflang" == "en" ]
then
  exit 6
fi
◇
```
Fragment referenced in 28b.
Uses: `naflang` 73.

### 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* 30c ⟩ ≡
```
if
  [ ! $alpino_installed ]
then
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
sicstus.tar.gz
  echo "ex-
port alpino_installed=0" >> /home/paul/projecten/pipelines/nlpp/installed_modules
fi
◇
```
Fragment referenced in 22a.

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* 31a ⟩ ≡
```
      export ALPINO_HOME=$modulesdir/Alpino
      ◇
```
Fragment referenced in 43e.
Defines: `ALPINO_HOME` Never used.

Remove the tarball when cleaning up:

⟨ *clean up* 31b ⟩ ≡
```
      rm -rf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
      sicstus.tar.gz
      ◇
```
Fragment defined by 13g, 15a, 31b, 79c.
Fragment referenced in 78d.

### 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* 31c ⟩ ≡
```
      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 31cde, 32abcd.
Fragment referenced in 22a.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 31d ⟩ ≡
```
      TREETAGSRC=tree-tagger-linux-3.2.tar.gz
      TREETAGSCRIPTS=tagger-scripts.tar.gz
      TREETAG_INSTALLSCRIPT=install-tagger.sh
      ◇
```
Fragment defined by 31cde, 32abcd.
Fragment referenced in 22a.
Uses: `install` 89a.

Parametersets:

⟨ *install the treetagger utility* 31e ⟩ ≡
```
      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 31cde, 32abcd.
Fragment referenced in 22a.

Download everything in the target directory:

⟨ *install the treetagger utility* 32a ⟩ ≡
```
      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 31cde, 32abcd.
Fragment referenced in 22a.

Run the install-script:

⟨ *install the treetagger utility* 32b ⟩ ≡
```
      chmod 775 $TREETAG_INSTALLSCRIPT
      ./$TREETAG_INSTALLSCRIPT
      ◇
```
Fragment defined by 31cde, 32abcd.
Fragment referenced in 22a.

Make the treetagger utilities available for everybody.

⟨ *install the treetagger utility* 32c ⟩ ≡
```
      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 31cde, 32abcd.
Fragment referenced in 22a.

Remove the tarballs:

⟨ *install the treetagger utility* 32d ⟩ ≡
```
      rm $TREETAGSRC
      rm $TREETAGSCRIPTS
      rm $TREETAG_INSTALLSCRIPT
      rm $DUTCHPARS_UTF_GZ
      rm $DUTCH_TAGSET
      rm $DUTCHPARS_2_GZ
      ◇
```
Fragment defined by 31cde, 32abcd.
Fragment referenced in 22a.

### 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* 33a ⟩ ≡
```
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* 33c ⟩
◇

Fragment referenced in 22a, 33d.

⟨ *install the timbl utility* 33b ⟩ ≡
```
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
```
⟨ *unpack ticcutils or timbl* 33c ⟩
◇

Fragment referenced in 22a, 33d.

⟨ *unpack ticcutils or timbl* 33c ⟩ ≡
```
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 33ab.
Uses: `install` 89a.

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

⟨ *re-install modules after the transplantation* 33d ⟩ ≡
⟨ *install the ticcutils utility* 33a ⟩
⟨ *install the timbl utility* 33b ⟩
◇

Fragment referenced in 90.

### 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* 34a ⟩ ≡
```
      cd $envdir
      tar -xzf $snapshotsocket/t_nlpp_resources/20160103_boost_1_54_bin.tgz
```
      ⋄

Fragment referenced in 22a.

### 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.

First, get the Spotlight model data that we need:

⟨ *download stuff* 34b ⟩ ≡

      ⟨ *need to wget* (34c `nl.tar.gz`,34d http://spotlight.sztaki.hu/downloads/archive/2014/nl.tar.gz ) 11b ⟩
      ⟨ *need to wget* (34e `en_2+2.tar.gz`,34f http://spotlight.sztaki.hu/downloads/archive/2014/en_2+2.tar.gz ) 11b ⟩
      ⟨ *need to wget* (34g `wikipedia-db.v1.tar.gz`,34h http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.

      ⋄

Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

⟨ *install the Spotlight server* 35a ⟩ ≡
```
cd $envdir
tar -xzf $snapshotsocket/t_nlpp_resources/spotlightnl.tgz
cd $envdir/spotlight
tar -xzf $snapshotsocket/t_nlpp_resources/nl.tar.gz
tar -xzf $snapshotsocket/t_nlpp_resources/en_2+2.tar.gz
```
◇
Fragment defined by 35ac.
Fragment referenced in 22a.

⟨ *get spotlight model ball* 35b ⟩ ≡
```
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 never referenced.

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

⟨ *install the Spotlight server* 35c ⟩ ≡
```
cd $envdir/spotlight
tar -xzf $snapshotsocket/$snapshotdirectory/wikipedia-db.v1.tar.gz
```
◇
Fragment defined by 35ac.
Fragment referenced in 22a.

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" 36a≡
        #!/bin/bash
        source /home/paul/projecten/pipelines/nlpp/env/bin/progenv
        ⟨ get commandline-arguments 27b ⟩
        ⟨ set default arguments for Spotlight 36b ⟩
        ◇
```
File defined by 36ac.


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

```
⟨ set default arguments for Spotlight 36b ⟩ ≡
        if
          [ "$spotlighthost" == "" ]
        then
          spotlighthost=130.37.53.33
        fi
        if
          [ "$spotlightport" == "" ]
        then
          if
            [ "$naflang" == "nl" ]
          then
              spotlightport=2060
          else
              spotlightport=2020
          fi
        fi
        ◇
```
Fragment referenced in 36a.
Uses: naflang 73.


```
"../bin/check_start_spotlight" 36c≡
        ⟨ check listener on host, port (36d $spotlighthost,36e $spotlightport ) 37d ⟩
        if
          [ $spotlightrunning -ne 0 ]
        then
          if
            [ ! "$spotlighthost" == "localhost" ]
          then
            export spotlighthost="localhost"
            ⟨ check listener on host, port (36f $spotlighthost,36g $spotlightport ) 37d ⟩
          fi
        fi
        if
          [ $spotlightrunning -ne 0 ]
        then
          ⟨ start the Spotlight server on localhost 39a, . . . ⟩
        fi
        echo $spotlighthost:$spotlightport
        ◇
```
File defined by 36ac.

⟨ *make scripts executable* 37a ⟩ ≡
```
      chmod 775  ../bin/check_start_spotlight
```
      ◇
Fragment defined by 26d, 37a, 88c.
Fragment referenced in 88d.


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* 37b ⟩ ≡
```
      spothostport=`/home/paul/projecten/pipelines/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 46a, 61b.
Uses: `naflang` 73, `print` 82b.


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

⟨ *get spotlight language parameters* 37c ⟩ ≡
```
      if
        [ "$naflang" == "nl" ]
      then
         spotlightport=2060
      else
        spotlightport=2020
      fi
```
      ◇
Fragment never referenced.
Uses: `naflang` 73.


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* 37d ⟩ ≡
```
      exec 6<>/dev/tcp/@1/@2 2>/dev/null
      spotlightrunning=$?
      exec 6<&-
      exec 6>&-
```
      ◇
Fragment referenced in 36c, 40a.


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* 38a ⟩ ≡
```
    ⟨ test whether spotlighthost runs (38b $spotlighthost ) 38e ⟩
    if
      [ ! $spotlightrunning ]
    then
      if
        [ "$spotlighthost" != "localhost" ]
      then
        export spotlighthost=localhost
        ⟨ test whether spotlighthost runs (38c $spotlighthost ) 38e ⟩
      fi
    fi
    if
      [ ! $spotlightrunning ]
    then
      ⟨ start the Spotlight server on localhost 39a, … ⟩
      ⟨ test whether spotlighthost runs (38d $spotlighthost ) 38e ⟩
    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* 38e ⟩ ≡
```
    exec 6<>/dev/tcp/@1/2060
    if
      [ $? -eq 0 ]
    then
      export spotlightrunning=0
    else
      spotlightrunning=
    fi
    exec 6<&-
    exec 6>&-
    ◇
```
Fragment referenced in 38a.

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* 39a ⟩ ≡

```
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 39ab.
Fragment referenced in 36c, 38a.
Uses: `naflang` 73.

⟨ *start the Spotlight server on localhost* 39b ⟩ ≡

```
local oldd=`pwd`
cd /home/paul/projecten/pipelines/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* 40a ⟩
```
  $envbindir/sematree release spotlock
else
```
  ⟨ *wait until the spotlight server is up or faulty* 40a ⟩
```
fi
cd $oldd
```
◇

Fragment defined by 39ab.
Fragment referenced in 36c, 38a.

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* 40a ⟩ ≡

```
      trial=0
      maxtrials=12
      while
        trial=$((trial+1))
        ⟨ check listener on host, port (40b $spotlighthost,40c $spotlightport ) 37d ⟩
        [ $spotlightrunning -ne 0 ] && [ $trial -le $maxtrials ]
      do
        sleep 10
      done
      if
        [ $spotlightrunning -ne 0 ]
      then
       export spotlighthost="none"
      fi
      ◇
```
Fragment referenced in 39b.

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* 40d ⟩ ≡

```
      cd $envdir/python
      git clone https://github.com/cltl/VUA_pylib.git
      ◇
```
Fragment referenced in 22a.

### 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.

⟨ *download stuff* 40e ⟩ ≡

```
      ⟨ need to wget (40f svm_light.tar.gz,40g http://download.joachims.org/svm_light/current/svm_light.tar.gz )
      ◇
```
Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

Installation goes like this:

⟨ *install SVMLight* 41a ⟩ ≡
```
      tempdir=`mktemp -d -t SVMlight.XXXXXX`
      cd $tempdir
      #tar -xzf $snapshotsocket/$snapshotdirectory/m4_SVMlightball
      wget http://download.joachims.org/svm_light/current/svm_light.tar.gz
      tar -xzf svm_light.tar.gz
      make
      cp svm_classify /home/paul/projecten/pipelines/nlpp/env/bin/
      cp svm_learn /home/paul/projecten/pipelines/nlpp/env/bin/
      cd /home/paul/projecten/pipelines/nlpp
      rm -rf $tempdir
      ◇
```
Fragment referenced in 22a.
Uses: `svm_classify` 70b, `svm_learn` 70b.

## 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* 41b ⟩ ≡
```
      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 23a.

*Script*   The script runs the tokenizerscript.

`"../bin/tok"` 41c≡
```
      ⟨ start of module-script (41d $jarsdir ) 28b ⟩
      JARFILE=$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* 42a ⟩ ≡
```
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* 42g ⟩
⟨ *compile the topic-tool jar* 43a ⟩
```
cd $modulesdir
rm -rf $tempdir
```
     ◇
Fragment referenced in 23a.

The two zip-balls `en-eurovoc-1.0.zip` and `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.

⟨ *download stuff* 42b ⟩ ≡

     ⟨ *need to wget* (42c `en-eurovoc-1.0.zip`,42d `http://optima.jrc.it/Resources/Eurovoc/indexing/en-eurovoc-1.0.`
     ⟨ *need to wget* (42e `nl-eurovoc-1.0.zip`,42f `http://optima.jrc.it/Resources/Eurovoc/indexing/nl-eurovoc-1.0.`
     ◇
Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

⟨ *install the jex resources and libraries* 42g ⟩ ≡
```
moddir=$modulesdir/ixa-pipe-topic
cd $moddir
mkdir -p jex/en
mkdir -p jex/nl
mkdir -p lib
cd $tempdir
unzip -q $snapshotsocket/$snapshotdirectory/en-eurovoc-1.0.zip
unzip -q $snapshotsocket/$snapshotdirectory/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 42a.

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* 43a ⟩ ≡

```
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 42a.
Uses: `install` 89a.

*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"` 43b≡

```
⟨ start of module-script (43c ixa-pipe-topic ) 28b ⟩
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* 43d ⟩ ≡

```
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=d5f002605d7c06545f24c84386342b79e5cb9c86
⟨ install from github 10a ⟩
cd $modulesdir/morphosyntactic_parser_nl
git checkout d5f002605d7c06545f24c84386342b79e5cb9c86
```
◇

Fragment referenced in 23a.

*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"` 43e≡

```
⟨ start of module-script (43f morphosyntactic_parser_nl ) 28b ⟩
⟨ get the mor time-out parameter 44a ⟩
⟨ set alpinohome 31a ⟩
cat | python $MODDIR/core/morph_syn_parser.py $timeoutarg
```
◇

Use getopts to read the **-t** option.

⟨ *get the mor time-out parameter* 44a ⟩ ≡

```
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 43e.

### 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* 44b ⟩ ≡

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

Fragment referenced in 23a.

*Script*

"../bin/pos" 44c≡

⟨ *start of module-script* ( 44d **EHU-pos.v30** ) 28b ⟩
```
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* 45a ⟩ ≡

```
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 23j.

*Script*

"../bin/constpars" 45b≡

⟨ *start of module-script* (45c `EHU-parse.v30` ) 28b ⟩
```
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

The NED-reranker is not open-source yet. It must be retrieved from the snapshot.

*Module*

⟨ *install the* NED-*reranker module* 45d ⟩ ≡
```
cd $modulesdir
pip install SPARQLWrapper
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-popen-nedreranker.v30.tgz
```
◇

Fragment referenced in 23j.
Uses: `install` 89a.

*Script*

"../bin/nedrer" 45e≡

⟨ *start of module-script* (45f `VUA-popen-nedreranker.v30` ) 28b ⟩
```
cd $MODDIR
python $MODDIR/domain_model.py
```
◇

### 5.6.7 Wikify module

*Module* Wikify is still obtained from the snapshot. Make sure that the jar of the module is accessable for everybody.

⟨ *install the wikify module* 45g ⟩ ≡
```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-wikify.v30.tgz
chmod 775 $modulesdir/EHU-wikify.v30/ixa-pipe-wikify-1.2.1.jar
```
◇

Fragment referenced in 23j.

*Script*   The Wikify module needs DBpedia to generate "markables".

```
"../bin/wikify" 46a≡
      ⟨ start of module-script (46b EHU-wikify.v30 ) 28b ⟩
      ⟨ find a spotlightserver or exit 37b ⟩
      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 46c ⟩ ≡
      ◇
```
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" 46d≡
      ⟨ start of module-script (46e EHU-ukb.v30 ) 28b ⟩
      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* 47a ⟩ ≡
```
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 24a.

*Script*

"../bin/ewsd" 47b≡
⟨ *start of module-script* (47c VUA-ims-wsd.v30 ) 28b ⟩
⟨ *set up Java 1.6* 15d ⟩
```
#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 English 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.

Anyway, we need a script that starts the server if it is not already running. We must be aware that in some applications a bunch of parallel processes may be running this script at the same time, which causes synchronisation errors. So do as follows:

1.     Check whether the server is in the air (listening on its port). If that is the case, we are ready.
2.     Otherwise, gain exclusive permission to proceed. (**P** operation)
3.     Check whether the "pidfile" (file that contains the process-id of the server) exists.
4.     If the pidfile does not exist, start the server (and write its pid in the pidfile). Give up exclusive permission (**V** operation). We are ready.
5.     Otherwise, if the pid-file exists, check whether a process with the given pid is running.
6.     If such a process is running, the server has been started, but it is may not yet be up. Give up exclusive permission (**V** operation). We are ready.
7.     If no running process has the given pid, the server may have been aborted. Start the server. Give up exclusive permission. We are ready.

The following scheme shows the process:

**Server up?**
yes: **ready**
no: **P**
    **PID-file exists?**
    no:  start server
        **V**

```
            ready
    yes: process with PID exists?
        yes: V
              ready
        no:  start server
              V
              ready
```

Set a default location or the directory for the pidfile, that can be overridden by user-set variable.

`"../env/bin/progenv"` 48a≡

```
    if
      [ -z ${eSRL_piddir+x} ]
    then
      export eSRL_piddir=/home/paul/projecten/pipelines/nlpp/env/etc/srlpid
    fi
    ◇
```
File defined by 8a, 13b, 48a.

⟨ *start EHU SRL server if it isn't running* 48b ⟩ ≡

```
    pidFile=$eSRL_piddir/SRLServer.pid
    mkdir -p $eSRL_piddir
    portInfo=$(nmap -p 5005 localhost | grep open)
    if
      [ -z "$portInfo" ]
    then
      sematree acquire esrl_lock
      if
        [ ! -e "$pidFile" ]
      then
          start_eSRL_server
      else
        pid=`cat $pidFile`
        ps -p $pid > /dev/null
        result=$?
        if
          [ $result -ne 0 ]
        then
         start_eSRL_server
        fi
      fi
      sematree release esrl_lock
    fi
    ◇
```
Fragment referenced in 50a.

⟨ *function to start EHU SRL server* 49a ⟩ ≡

```
function start_eSRL_server {
  pidFile=$eSRL_piddir/SRLServer.pid
  java -Xms2500m -cp $MODDIR/IXA-EHU-srl-
3.0.jar ixa.srl.SRLServer en &> /dev/null &
  pid=$!
  echo $pid > $pidFile
}
```

◇

Fragment referenced in 50a.

When the server is up and running, it serves port 5005. So when we know that the server has been started up, let us wait until that port is served. When it takes too long, give up.

⟨ *wait for SRL server* 49b ⟩ ≡

```
nr_of_trials=0
maxtrials=20
while
  portInfo=$(nmap -p 5005 localhost | grep open)
  [ -z "$portInfo" ] && [ $nr_of_trials -le $maxtrials ]
do
  sleep 10
  nr_of_trials=$((nr_of_trials+1))
done
if
  [ -z "$portInfo" ]
then
  echo "eSRL server does not start."  >&2
fi
```
◇

Fragment referenced in 50c.

*Module*

⟨ *install the srl-server module* 49c ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-srl-server.tgz
cd EHU-srl-server
mkdir -p /home/paul/projecten/pipelines/nlpp/env/etc/srlpid
chmod o+rwx /home/paul/projecten/pipelines/nlpp/env/etc/srlpid
```
◇

Fragment referenced in 24a.

*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" 50a≡
```
#!/bin/bash
source /home/paul/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
MODDIR=$modulesdir/EHU-srl-server
```
⟨ *function to start EHU SRL server* 49a ⟩
⟨ *start EHU SRL server if it isn't running* 48b ⟩
◇

"../bin/stop_eSRL" 50b≡
```
#!/bin/bash
source /home/paul/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
pidFile=$eSRL_piddir/SRLServer.pid
if
  [ -e "$pidFile" ]
 then
  kill `cat $pidFile`
  rm $pidFile
 fi
```
◇

"../bin/eSRL" 50c≡
⟨ *start of module-script* (50d EHU-srl-server ) 28b ⟩
```
/home/paul/projecten/pipelines/nlpp/bin/start_eSRL
```
⟨ *wait for SRL server* 49b ⟩
```
java -Xmx1000m -cp $MODDIR/IXA-EHU-srl-3.0.jar ixa.srl.SRLClient en
```
◇

⟨ *start EHU SRL server* 50e ⟩ ≡
```
pidFile=$eSRL_piddir/SRLServer.pid
java -Xms2500m -cp $MODDIR/IXA-EHU-srl-3.0.jar ixa.srl.SRLServer en &> /dev/null &
pid=$!
echo $pid > $pidFile
```
◇
Fragment never referenced.

### 5.6.11   SRL Dutch nominals

*Module*

⟨ *install the srl-dutch-nominals module* 50f ⟩ ≡
```
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* 10a ⟩
```
cd $modulesdir/vua-srl-dutch-nominal-events
```
◇
Fragment referenced in 24a.

*Script*

```
"../bin/srl-dutch-nominals" 51a≡
      ⟨ start of module-script (51b vua-srl-dutch-nominal-events ) 28b ⟩
      cd $MODDIR
      cat | python $MODDIR/vua-srl-dutch-additional-roles.py
      ◇
```

### 5.6.12  FBK-time module

The FBK-time module is obtained from a tarball that has been ripped from the Newsreader server.
Note: on june 15, 2016 jar file `TimeProNorm_v2.6.jar` has been added to the `lib` subdirectory.

Anne-Lyse Minard wrote in an e-mail on june 10, 2016:

```
For the following example:
"*will* have a collective chance to discuss the British demands at an EU summit on October 15-16"

The day "16" is considered as the year, it is why it is annotated as 2016-10-15.
I cannot do much because "15-16" is considered as one token. In TimeML it would have been annotat
<TIMEX3 value="2015-10-15">October 15</TIMEX3>-<TIMEX3 value="2015-10-16">16</TIMEX3>.
But in NAF as the annotation is done at the token level this is not possible.

What I have done to solve these types of errors is to consider only "October 15" and so to normal

A new version of TimeProNorm can be download from https://github.com/alminard/TimeProNorm/blob/ma

I don't have time now to test the changes I have done (I have just run it on 2 files).
How urgent is it for you to process the data?
```

*Module*

⟨ *install the FBK-time module* 51c ⟩ ≡
```
      cd $modulesdir
      tar -xzf $snapshotsocket/t_nlpp_resources/20160616_FBK-time.v30.tgz
      ◇
```
Fragment referenced in 24j.

*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" 52a≡
      ⟨ start of module-script (52b FBK-time.v30 ) 28b ⟩
      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 (52c Java: $JAVACLASSPATH:$JAVAMODULE ) 53a ⟩
      #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 \
        >> $TIMEPRONORMIN
      ⟨ stop on error (52d yamcha ) 53a ⟩
      JAVACLASSPATH="lib/scala-library.jar:lib/timenorm-0.9.1-SNAPSHOT.jar"
      JAVACLASSPATH=$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 (52e Java: $JAVACLASSPATH:$JAVAMODULE ) 53a ⟩
      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 (52f Java: $JAVACLASSPATH:$JAVAMODULE ) 53a ⟩
      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 (52g Java: $JAVACLASSPATH:$JAVAMODULE ) 53a ⟩
      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* 53a ⟩ ≡
```
result=$?
if
  [ $result -ne 0 ]
then
  cd $MODDIR
  echo Error: @1 >&2
  rm -rf $timdir
  exit $result
fi
```
◇

Fragment referenced in 52a.

### 5.6.13 FBK-temprel module

*Module*

⟨ *install the FBK-temprel module* 53b ⟩ ≡
```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-temprel.v30.tgz
```
⟨ *repair FBK-\*rel's run.sh.hadoop* (53c `FBK-temprel.v30` ) 53d ⟩
◇

Fragment referenced in 24j.

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* 53d ⟩ ≡
```
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 53b, 54c.

*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" 54a≡
      ⟨ start of module-script (54b FBK-temprel.v30 ) 28b ⟩
      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 54c ⟩ ≡
      cd $modulesdir
      tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-causalrel.v30.tgz
      ⟨ repair FBK-*rel's run.sh.hadoop (54d FBK-causalrel.v30 ) 53d ⟩
      ◇
```
Fragment referenced in 24j.

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 54e ⟩ ≡
      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" 55a≡
      ⟨ start of module-script (55b FBK-causalrel.v30 ) 28b ⟩
      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 modules

At this moment there is not yet a single factuality modules that server Dutch as well as English.
In spite of it's name, the multilingual_factuality module serves only Dutch on production-quality
level. For English, we have the VUA-factuality module.

*Modules*

⟨ *install the factuality modules* 55c ⟩ ≡
```
      MODNAM=vua_factuality
      DIRN=vua_factuality
      GITU=https://github.com/cltl/vua_factuality.git
      GITC=58fae1dc27d503642f4899460a3481417f044e95
```
      ⟨ *install from github* 10a ⟩
```
      MODNAM=multilingual_factuality
      DIRN=multilingual_factuality
      GITU=https://github.com/cltl/multilingual_factuality
      GITC=cbad4849b7d4421790589b018a9073d646ca45e3
```
      ⟨ *install from github* 10a ⟩

      ◇
Fragment referenced in 24j.

*Script*

```
"../bin/factuality" 55d≡
      ⟨ start of module-script (55e LATER ) 28b ⟩
      if
        [ "$naflang" == "nl" ]
      then
        MODDIR=$modulesdir/multilingual_factuality
        ⟨ run dutch factuality 56b ⟩
      else
        MODDIR=$modulesdir/vua_factuality
        ⟨ run english factuality 56a ⟩
      fi


      ◇
```

⟨ *run english factuality* 56a ⟩ ≡
```
rootDir=${MODDIR}
tmpDir=$(mktemp -d -t factuality.XXXXXX)
export PATH=$PATH:${rootDir}:.
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/paul/projecten/pipelines/nlpp/env/lib/
cd $rootDir
python ${rootDir}/vua_factuality_naf_wrapper.py -
t /home/paul/projecten/pipelines/nlpp/env/bin/timbl -p ${rootDir} ${tmpDir}/
rm -rf $tmpDir >& /dev/null
```
◇
Fragment referenced in 55d.
Uses: LD_LIBRARY_PATH 17e, PATH 7e.

To run the Dutch factuality is very simple. It is a Python script that finds out its own location.

⟨ *run dutch factuality* 56b ⟩ ≡
```
cd $MODDIR/feature_extractor
cat | python ./rule_based_factuality.py
```
◇
Fragment referenced in 55d.

### 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* 56c ⟩ ≡
```
cd $modulesdir
tar -xzf /home/paul/projecten/pipelines/t_nlpp_resources/20151220_EHU-
corefgraph.v30.tgz
```
◇
Fragment referenced in 25a.

*Script*

```
"../bin/coreference-base" 56d≡
    ⟨ start of module-script (56e EHU-corefgraph.v30 ) 28b ⟩
    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* 57a ⟩ ≡
  ⟨ *compile the nerc jar* 57b ⟩
  ⟨ *get the nerc models* 57d ⟩


  ◇
Fragment referenced in 23j.


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* 57b ⟩ ≡
```
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 57a.


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* 57c ⟩ ≡
```
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` 73.


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* 57d ⟩ ≡
```
cd $modulesdir
tar -p -
xzf /home/paul/projecten/pipelines/t_nlpp_resources/20160301_nerc_models.tgz
```
  ◇
Fragment referenced in 57a.

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

`"../bin/nerc_conll02"` 58a≡
      ⟨ *start of module-script* (58b `m4_nerc_nl_dir` ) 28b ⟩
      `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"` 58c≡
      ⟨ *start of module-script* (58d `nerc-models` ) 28b ⟩
      `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* 58e ⟩ ≡
      `MODNAM=wsd`
      `DIRN=svm_wsd`
      `GITU=https://github.com/cltl/svm_wsd.git`
      `GITC=030043903b42f77cd20a9b2443de137e2efe8513`
      ⟨ *install from github* 10a ⟩
      `cd $modulesdir/svm_wsd`
      ⟨ *install svm lib* 59a ⟩
      ⟨ *download svm models* 59b ⟩

      ◇
Fragment referenced in 25a.


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

⟨ *install svm lib* 59a ⟩ ≡

```
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 58e.

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

⟨ *download svm models* 59b ⟩ ≡

```
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 58e.

*Script*

```
"../bin/wsd" 59c≡
    ⟨ start of module-script (59d svm_wsd ) 28b ⟩
    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* 59e ⟩ ≡

```
cd $modulesdir
tar -xzf $snapshotsocket/t_nlpp_resources/lu2synset.tgz
    ◇
```
Fragment referenced in 25h.

*Script*

```
"../bin/lu2synset" 60a≡
       ⟨ start of module-script (60b lexicalunitconvertor ) 28b ⟩
       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* 60c ⟩ ≡
       ⟨ *put spotlight jar in the Maven repository* 61a ⟩
       MODNAM=ned
       DIRN=ixa-pipe-ned
       GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
       GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
       ⟨ *install from github* 10a ⟩
       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 23j.

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. The Dutch data in `nl.tar.gz` seems to be needed as
well. We already downloaded that resource for Spotlight itself, so we do not download that again.

⟨ *download stuff* 60d ⟩ ≡

       ⟨ *need to wget* (60e **dbpedia-spotlight-0.7.jar**,60f http://spotlight.sztaki.hu/downloads/archive/2014/dbpedia
       ◇
Fragment defined by 11ac, 14c, 19b, 34b, 40e, 42b, 60d.
Fragment referenced in 9c.

⟨ *put spotlight jar in the Maven repository* 61a ⟩ ≡
```
echo Put Spotlight jar in the Maven repository.
tempdir=`mktemp -d -t simplespot.XXXXXX`
cd $tempdir
cp $snapshotsocket/$snapshotdirectory/dbpedia-spotlight-0.7.jar .
tar -xzf $snapshotsocket/$snapshotdirectory/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 60c.
Uses: install 89a.

*Script* NED needs to contact a Spotlight-server.

`"../bin/ned"` 61b≡
```
⟨ start of module-script (61c ) 28b ⟩
ROOT=$piperoot
JARDIR=$jarsdir
⟨ find a spotlightserver or exit 37b ⟩
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 Dark entity relinker

The "Dark Entity Relinker" tries to link "Dark entities" (named entities that have not been recognized) to the link of a known entity with a similar name structure that has been found in the same text.

*Module* Install the module from Github.

⟨ *install the de-link module* 61d ⟩ ≡
```
cd $modulesdir
git clone https://github.com/cltl/entity-relink-pipeline.git
cd entity-relink-pipeline
git checkout 90d7b64840dda8ce3baf744e140891a6f8419052
```
◇

Fragment referenced in 23j.

*script*

```
"../bin/derel" 62a≡
      ⟨ start of module-script (62b entity-relink-pipeline ) 28b ⟩
      cd $MODDIR/
      cat | python relinkDarkEntities.py
      ◇
```

### 5.6.22  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* 62c ⟩ ≡
```
      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 25a.
Uses: `install` 89a.

### Scripts   The "onto" (predicatematrix) script:

```
"../bin/onto" 62d≡
      ⟨ start of module-script (62e OntoTagger ) 28b ⟩
      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" 62f≡
      ⟨ start of module-script (62g OntoTagger ) 28b ⟩
      cd $MODDIR/scripts
      cat | $MODDIR/scripts/srl-framenet-
      tagger.sh | gawk '/^frameMap.size()/ {next}; {print}'


      ◇
```

The "nomevent" script:

```
"../bin/nomevent" 63a≡
      ⟨ start of module-script (63b OntoTagger ) 28b ⟩
      cd $MODDIR/scripts
      cat | $MODDIR/scripts/nominal-events.sh
      ◇
```

### 5.6.23  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 63c ⟩ ≡
      moduledir=/home/paul/projecten/pipelines/nlpp/modules/ixa-pipe-time
      heideltemp=`mktemp -d -t heidel.XXXXXX`
      ⟨ clone the heideltime wrapper 63d ⟩
      ⟨ put miscellaneous stuff in the heideltime module 63f ⟩
      ⟨ compile the heideltime wrapper 64a ⟩
      rm -rf $heideltemp
      ◇
```
Fragment referenced in 25h.

```
⟨ clone the heideltime wrapper 63d ⟩ ≡
      MODNAM=heideltime
      DIRN=ixa-pipe-time
      GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
      GITC=0fd3b5bd4c9a82b1624928a487b3a963a266f10c
      ⟨ install from github (63e ixa-pipe-time ) 10a ⟩
      ◇
```
Fragment referenced in 63c.

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 miscellaneous stuff in the heideltime module 63f ⟩ ≡
      cd $heideltemp
      tar -xzf $snapshotsocket/t_nlpp_resources/20151123_antske_heideltime_stuff.tgz
      cp antske_heideltime_stuff/config.props $moduledir/
      wget http://ixa2.si.ehu.es/~jibalari/jvntextpro-2.0.jar
      mv jvntextpro-2.0.jar $moduledir/lib/
      git clone https://github.com/carchrae/install-to-project-repo.git
      ◇
```
Fragment referenced in 63c.
Uses: install 89a.

Compile the Heideltime wrapper according to the instruction on Github.

⟨ *compile the heideltime wrapper* 64a ⟩ ≡

```
cd /home/paul/projecten/pipelines/nlpp/modules/$DIRN
python $heideltemp/install-to-project-repo/install-to-project-repo.py
mvn clean install
```
◇

Fragment referenced in 63c.
Uses: install 89a.

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* 64b ⟩ ≡

```
        ⟨ remove outdated heideltime jars 64c ⟩
cd /home/paul/projecten/pipelines/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 never referenced.
Uses: install 89a.

⟨ *remove outdated heideltime jars* 64c ⟩ ≡

```
rm -rf /home/paul/projecten/pipelines/nlpp/modules/$DIRN/repo
mkdir -p /home/paul/projecten/pipelines/nlpp/modules/$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 64b.

*Script*

`"../bin/heideltime"` 64d≡

```
        ⟨ start of module-script (64e ixa-pipe-time ) 28b ⟩
MODDIR=$modulesdir/ixa-pipe-time
cd $MODDIR
iconv -t utf-8//IGNORE | java -Xmx1000m -jar target/ixa.pipe.time.jar -
m lib/alpino-to-treetagger.csv -c config.props
```
◇

### 5.6.24 Semantic Role labelling

*Module*

⟨ *install the srl module* 65a ⟩ ≡
```
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=675d22d361289ede23df11dcdb17195f008c54bf
```
⟨ *install from github* 10a ⟩
                        ◇

Fragment referenced in 25h.

*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"` 65b≡
⟨ *start of module-script* (65c `vua-srl-nl` ) 28b ⟩
```
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 65bdefg.

Create a feature-vector.

`"../bin/srl"` 65d≡
```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
```
                        ◇

File defined by 65bdefg.

Run the trained model on the feature-vector.

`"../bin/srl"` 65e≡
```
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 65bdefg.

Insert the SRL values into the NAF file.

`"../bin/srl"` 65f≡
```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
```
                        ◇

File defined by 65bdefg.

Clean up.

`"../bin/srl"` 65g≡
```
rm -rf $TEMPDIR
```
                        ◇

File defined by 65bdefg.

5.6.25  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* 66a ⟩ ≡
```
      pip install rdflib
```
      ◇

Fragment defined by 19a, 66a.
Fragment referenced in 15e.
Defines: `rdflib` Never used.
Uses: `install` 89a.

⟨ *install the post-SRL module* 66b ⟩ ≡
```
      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 25q.

*Script*

"../bin/postsrl" 66c≡
```
      ⟨ start of module-script (66d vua-srl-postprocess ) 28b ⟩
      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
```
      ◇

5.6.26  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* 67a ⟩ ≡

```
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 25h.

### Script

"../bin/evcoref" 67b≡

```
⟨ start of module-script (67c vua-eventcoreference_v2 ) 28b ⟩
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

◇
```

### 5.6.27  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* 67d ⟩ ≡

```
MODNAM=dbpedia_ner
DIRN=dbpedia_ner
GITU=https://github.com/PaulHuygen/dbpedia_ner.git
GITC=ab1dcbd860f0ff29bc979f646dc382122a101fc2
⟨ install from github 10a ⟩
◇
```

Fragment referenced in 25q.

*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"` 68a≡
⟨ *start of module-script* (68b `dbpedia_ner` ) 28b ⟩
```
cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
url http://$spotlighthost:2060/rest/candidates
```
◇

### 5.6.28  GetSemfromNaf

The `naf2sem` module is a application that is hidden in the EventCoreference jar ([https://github.com/cltl/EventCoreference](https://github.com/cltl/EventCoreference)). It uses resources from `https://github.com/cltl/vua-resources`, that has been installed by the ontotagger installer.

*Module*

⟨ *install naf2sem* 68c ⟩ ≡
```
n2sdir=`mktemp -d -t n2s.XXXXXX`
cd $n2sdir
MODNAM=EventCoreference
GITC=52765c55354fda9d5d3046e91d4b0f99898b014b
git clone https://github.com/cltl/EventCoreference.git
cd $MODNAM
git checkout $GITC
set -e
mvn clean
mvn install
jarfile=EventCoreference-v3.1.2-jar-with-dependencies.jar
if
  [ -d "target/$jarfile" ]
then
  mv target/$jarfile $jarsdir
else
  echo "Could not generate EventCoreference-v3.1.2-jar-with-dependencies.jar
  exit 4
fi
cd "$modulesdir"
rm -rf $n2sdir
```

◇
Fragment never referenced.
Uses: `install` 89a.

*Script*   The script is not yet correct.

```
"../bin/naf2sem" 69a≡
      ⟨ start of module-script (69b dbpedia_ner ) 28b ⟩
      JAR=$jarsdir/EventCoreference-v3.1.2-jar-with-dependencies.jar
      RESOURCES="$modulesdir/vua-resources"
      DATA="../data2"

      java -Xmx2000m -cp "$LIB/EventCoreference-v3.1.2-jar-with-
      dependencies.jar" eu.newsreader.eventcoreference.naf.GetSemFromNafFolder --naf-
      folder "$DATA" --extension ".naf" --verbose --project dasym --all --
      ili "$RESOURCES/ili.ttl.gz" --perspective --source-frames "$RESOURCES/source-
      nl.txt" --non-entities

      # --eurovoc-en "$RESOURCES/mapping_eurovoc_skos.csv.gz"
      # --no-doc-time
      # --no-context-time
      # --non-entities


      ◇
```

## 5.6.29 Opinion miner

Get `opinion-miner_deluxePP` from Github. The repository contains an install script `install_me` that installs the following:

- KafNafParserPy (we have this already);
- The "Conditional Random Fields" program, that is included in the tarball;
- The `SVM_light` utility that we installed in section 5.5.10.
- Trained models (hotels, news).
- Polarity-models.

So, we need to do the following:

- Get the module from the Github repo.
- Install the included CRF program.
- Link the binaries of SVM to a place where the opinion-miner expects it.
- Download and install the models.

*Module*   Install the module from Github.

⟨ *install the opinion-miner* 69c ⟩ ≡
```
      MODNAM=opinion_miner_deluxePP
      DIRN=opinion_miner_deluxePP
      GITU=https://github.com/rubenIzquierdo/opinion_miner_deluxePP
      GITC=40a714c9823d439262caa29f3a14c5ce2be91241
      ⟨ install from github 10a ⟩
      ◇
```
Fragment defined by 69c, 70abcd.
Fragment referenced in 26a.

Install the CRF uitilty:

⟨ *install the opinion-miner* 70a ⟩ ≡
```
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
rm -rf $moduledir/crf_lib/CRF++-0.58.tar.gz
cd CRF++-0.58
./configure --prefix=$envdir
make
make install
echo "PATH_TO_CRF_TEST='$envbindir/crf_test'" > $moduledir/path_crf.py
cd $moduledir
rm -rf $crfdir
      ◇
```
Fragment defined by 69c, 70abcd.
Fragment referenced in 26a.
Uses: `install` 89a.

The opinion-miner expects the SVM-light executables in a subdirectory `svm_light`.

⟨ *install the opinion-miner* 70b ⟩ ≡
```
cd $moduledir
mkdir $moduledir/svm_light
ln -
s /home/paul/projecten/pipelines/nlpp/env/bin/svm_classify $moduledir/svm_light/
ln -s /home/paul/projecten/pipelines/nlpp/env/bin/svm_learn $moduledir/svm_light/
      ◇
```
Fragment defined by 69c, 70abcd.
Fragment referenced in 26a.
Defines: `svm_classify` 41a, `svm_learn` 41a.

Download the trained models.

⟨ *install the opinion-miner* 70c ⟩ ≡
```
##Download the models
cd $moduledir
echo Downloading the trained models.
tar -xzf $snapshotsocket/t_nlpp_resources/models_opinion_miner_deluxePP.tgz
      ◇
```
Fragment defined by 69c, 70abcd.
Fragment referenced in 26a.

⟨ *install the opinion-miner* 70d ⟩ ≡
```
wget http://kyoto.let.vu.nl/~izquierdo/public/polarity_models.tgz
tar -xzf polarity_models.tgz
rm polarity_models.tgz
      ◇
```
Fragment defined by 69c, 70abcd.
Fragment referenced in 26a.
Defines: `polarity_models.tgz` Never used.

*Script*

```
"../bin/opinimin" 71a≡
      ⟨ start of module-script (71b opinion_miner_deluxePP ) 28b ⟩
      cd $MODDIR
      python tag_file.py -polarity -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 71c ⟩ ≡
      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 74b, 75a.
Defines: BIND 75b, moduleresult 74b, 75a, runmodule 72ab, 73.

**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* 72a ⟩ ≡
```
    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     derel                      derel.naf
    runmodule derel.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    factuality      fact.naf
    runmodule fact.naf    opinimin            opin.naf
    runmodule opin.naf    evcoref          $outfile
        ◇
```
Fragment never referenced.
Uses: runmodule 71c.


Similar for an English naf:

⟨ *annotate english document* 72b ⟩ ≡
```
      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     derel                      derel.naf
      runmodule derel.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 73.
Uses: runmodule 71c.


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* 73 ⟩ ≡

```
      naflang=`cat $infile | /home/paul/projecten/pipelines/nlpp/env/bin/langdetect.py`
      export naflang
      if
       [ "$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     derel                       derel.naf
      runmodule derel.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    factuality        fact.naf
      runmodule fact.naf    opinimin             opin.naf
      runmodule opin.naf    evcoref           $outfile
      else
         ⟨ annotate english document 72b ⟩
      fi
      ⋄
```

Fragment referenced in 74b, 75a.
Defines: naflang 27b, 30ab, 36b, 37bc, 39a, 41c, 43b, 55d, 57c, 58c, 67b.
Uses: runmodule 71c.

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* 74a ⟩ ≡

```
    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 74b.
Uses: nuweb 84d.


This is the test-script:

"../bin/test" 74b≡

```
    #!/bin/bash
    oldd=`pwd`
    ⟨ set variables for test/run script 75b ⟩
    workdir=$piperoot/test
    mkdir -p $workdir
    cd $workdir
    ⟨ get a testfile or die 74a ⟩
    ⟨ function to run a module 71c ⟩
    ⟨ annotate 73 ⟩
    if
      [ $moduleresult -eq 0 ]
    then
      echo Test succeeded.
    else
      echo Something went wrong.
    fi
    exit $moduleresult
    ◇
```
Uses: moduleresult 71c.


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" 75a≡
      #!/bin/bash
      oldd=`pwd`
      ⟨ set variables for test/run script 75b ⟩
      workdir=`mktemp -d -t nlpp.XXXXXX`
      cd $workdir
      cat >$workdir/$infile
      ⟨ function to run a module 71c ⟩
      ⟨ annotate 73 ⟩
      if
        [ $moduleresult -eq 0 ]
      then
        cat $outfile
      fi
      cd $oldd
      rm -rf $workdir
      exit $moduleresult
      ◇
```
Uses: moduleresult 71c.

```
⟨ set variables for test/run script 75b ⟩ ≡
      ROOT=/home/paul/projecten/pipelines/nlpp
      source /home/paul/projecten/pipelines/nlpp/env/bin/progenv
      BIND=$pipebin
      infile=in.naf
      outfile=out.naf
      ◇
```
Fragment referenced in 74b, 75a.
Uses: BIND 71c.

## 6.2 Logging

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

```
⟨ variables of install-modules 75c ⟩ ≡
      LOGLEVEL=1
      ◇
```
Fragment referenced in 22a.

```
⟨ logmess 75d ⟩ ≡
      if
        [ $LOGLEVEL -gt 0 ]
      then
       echo @1
      fi
      ◇
```
Fragment referenced in 9a, 10a, 76a.

## 6.3 Misc

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

**URL:** The URL from 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* 76a ⟩ ≡
```
    SUCCES=0
    cd $modulesdir
```
    ⟨ *move module* (76b $DIR ) 8b ⟩
```
    wget $URL
    SUCCES=$?
    if
       [ $SUCCES -eq 0 ]
    then
       tar -xzf $TARB
       SUCCES=$?
       rm -rf $TARB
    fi
    if
       [ $SUCCES -eq 0 ]
    then
```
       ⟨ *logmess* (76c Installed $DIR ) 75d ⟩
       ⟨ *remove old module* (76d $DIR ) 8c ⟩
```
    else
```
       ⟨ *re-instate old module* (76e $DIR ) 9a ⟩
```
    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 $ > \equiv$

```
     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 print-
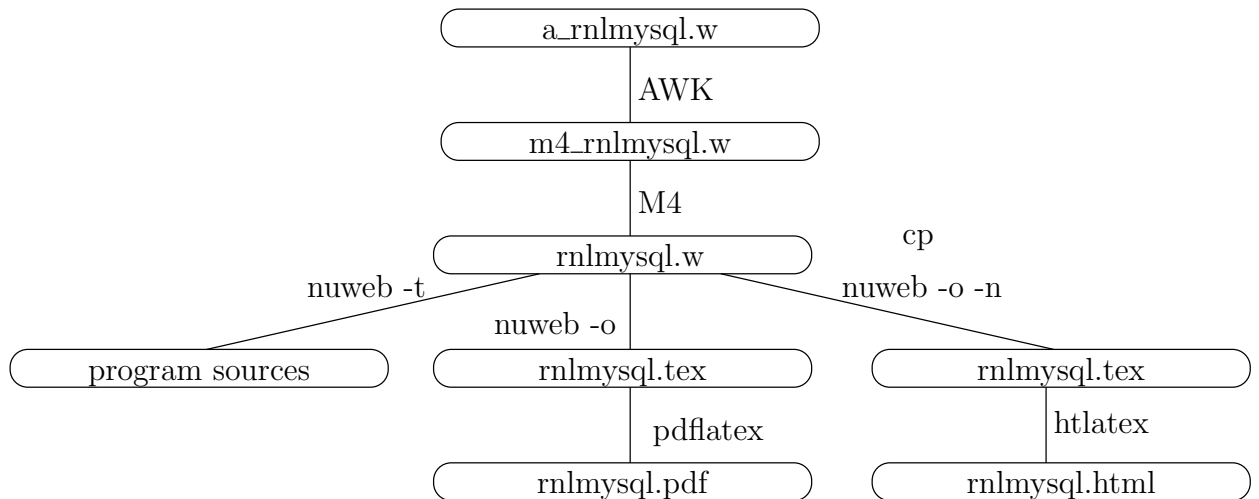


Figure 2: *Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.*

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

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

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

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

⟨ *parameters in Makefile* 78a ⟩ ≡
        `NUWEB=../env/bin/nuweb`
        ◇

Fragment defined by 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Uses: `nuweb` 84d.

## A.3   The Makefile for this project.

This chapter assembles the Makefile for this project.

`"Makefile"` 78b≡
        ⟨ *default target* 78c ⟩

        ⟨ *parameters in Makefile* 78a, … ⟩

        ⟨ *impliciete make regels* 81c, … ⟩
        ⟨ *expliciete make regels* 79b, … ⟩
        ⟨ *make targets* 78d, … ⟩
        ◇

The default target of make is `all`.

⟨ *default target* 78c ⟩ ≡
        `all  :` ⟨ *all targets* 78e ⟩
        `.PHONY : all`


        ◇

Fragment referenced in 78b.
Defines: `all` 69a, `PHONY` 82a.

⟨ *make targets* 78d ⟩ ≡
        `clean:`
                ⟨ *clean up* 13g, … ⟩


        ◇

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.

The default is, to install nlpp.

⟨ *all targets* 78e ⟩ ≡
        `install`◇

Fragment referenced in 78c.
Uses: `install` 89a.

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* 79a ⟩ ≡
```
      .SUFFIXES: .pdf .w .tex .html .aux .log .php
```

◇

Fragment defined by 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 82b.

## 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* 79b ⟩ ≡
```
      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 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.
Uses: `nuweb` 84d.

⟨ *clean up* 79c ⟩ ≡
```
      rm -rf ../nuweb-1.58
```
◇

Fragment defined by 13g, 15a, 31b, 79c.
Fragment referenced in 78d.
Uses: `nuweb` 84d.

⟨ *expliciete make regels* 79d ⟩ ≡
```
      ../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 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.
Uses: `nuweb` 84d.

## 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* 80a ⟩ ≡
```
      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 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.
Uses: `print` 82b.

### A.5.2  Run the M4 pre-processor

⟨ *expliciete make regels* 80b ⟩ ≡
```
      nlpp.w : m4_nlpp.w inst.m4
              m4 -P m4_nlpp.w > nlpp.w
```

        ⋄
Fragment defined by 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.

## 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* 81a ⟩ ≡
```
      FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Defines: FIGFILES 81b.

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* 81b ⟩ ≡
```
      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 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Defines: FIGFILENAMES Never used, PDFT_NAMES 83a, PDF_FIG_NAMES 83a, PST_NAMES Never used,
       PS_FIG_NAMES Never used.
Uses: FIGFILES 81a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 81c ⟩ ≡
```
      %.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 81c, 86a.
Fragment referenced in 78b.
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* 82a ⟩ ≡
```
        bibfile : nlpp.aux /home/paul/bin/mkportbib
                /home/paul/bin/mkportbib nlpp litprog


        .PHONY : bibfile
```
        ◇
Fragment defined by 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.
Uses: PHONY 78c.

A.6.3  Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 82b ⟩ ≡
```
        pdf : nlpp.pdf


        print : nlpp.pdf
                lpr nlpp.pdf


        view : nlpp.pdf
                evince nlpp.pdf

```
        ◇
Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Defines: pdf 79a, 83a, print 16a, 18b, 21c, 29b, 37b, 62f, 80a, 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* 83a ⟩ ≡
```
    nlpp.pdf : nlpp.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
            chmod 775 $(W2PDF)
            $(W2PDF) $*
```

◇

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Uses: pdf 82b, PDFT_NAMES 81b, PDF_FIG_NAMES 81b.

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* 83b ⟩ ≡
```
    ../nuweb/bin
```
◇
Fragment defined by 6, 7abc, 13c, 14bf, 17c, 83b.
Fragment referenced in 88b.
Uses: nuweb 84d.

⟨ *parameters in Makefile* 83c ⟩ ≡
```
    W2PDF=../nuweb/bin/w2pdf
```
◇
Fragment defined by 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Uses: nuweb 84d.

⟨ *expliciete make regels* 83d ⟩ ≡
```
    $(W2PDF) : nlpp.w $(NUWEB)
            $(NUWEB) nlpp.w
```
◇
Fragment defined by 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.

`"../nuweb/bin/w2pdf"` 83e≡
```
    #!/bin/bash
    # w2pdf -- compile a nuweb file
    # usage: w2pdf [filename]
    # 20170510 at 1359h: Generated by nuweb from a_nlpp.w
    NUWEB=../env/bin/nuweb
    LATEXCOMPILER=pdflatex
```
⟨ *filenames in nuweb compile script* 84b ⟩
⟨ *compile nuweb* 84a ⟩

◇
Uses: nuweb 84d.

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* 84a ⟩ ≡
```
        NUWEB=/home/paul/projecten/pipelines/nlpp/env/bin/nuweb
        ⟨ run the processors until the aux file remains unchanged 85a ⟩
        ⟨ remove the copy of the aux file 84c ⟩
        ◇
```
Fragment referenced in 83e.
Uses: `nuweb` 84d.

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* 84b ⟩ ≡
```
        nufil=$1
        trunk=${1%%.*}
        texfil=${trunk}.tex
        auxfil=${trunk}.aux
        oldaux=old.${trunk}.aux
        indexfil=${trunk}.idx
        oldindexfil=old.${trunk}.idx
        ◇
```
Fragment referenced in 83e.
Defines: `auxfil` 85a, 87ab, `indexfil` 85a, 87a, `nufil` 84d, 87ac, `oldaux` 84c, 85a, 87ab, `oldindexfil` 85a, 87a,
        `texfil` 84d, 87ac, `trunk` 84d, 87acd.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 84c ⟩ ≡
```
        rm $oldaux
        ◇
```
Fragment referenced in 84a, 86e.
Uses: `oldaux` 84b, 87a.

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* 84d ⟩ ≡
```
        $NUWEB $nufil
        $LATEXCOMPILER $texfil
        makeindex $trunk
        bibtex $trunk
        ◇
```
Fragment referenced in 85a.
Defines: `bibtex` 87cd, `makeindex` 87cd, `nuweb` 7d, 74a, 78a, 79bcd, 83bce, 84a, 85b, 86d.
Uses: `nufil` 84b, 87a, `texfil` 84b, 87a, `trunk` 84b, 87a.

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* 85a ⟩ ≡
```
      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* 84d ⟩
```
        if [ $LOOPCOUNTER -ge 10 ]
        then
          cp $auxfil $oldaux
        fi;
      done
```
        ◇

Fragment referenced in 84a.
Uses: `auxfil` 84b, 87a, `indexfil` 84b, `oldaux` 84b, 87a, `oldindexfil` 84b.

### 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* 85b ⟩ ≡
```
      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 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Uses: `nuweb` 84d.

Make the directory:

⟨ *expliciete make regels* 85c ⟩ ≡
```
      $(htmldir) :
              mkdir -p $(htmldir)
```

        ◇

Fragment defined by 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.

The rule to copy files in it:

⟨ *impliciete make regels* 86a ⟩ ≡
```
    $(htmldir)/% : % $(htmldir)
            cp $< $(htmldir)/
```

◇

Fragment defined by 81c, 86a.
Fragment referenced in 78b.

Do the work:

⟨ *expliciete make regels* 86b ⟩ ≡
```
    $(htmltarget) : $(htmlmaterial) $(htmldir)
            cd $(htmldir) && chmod 775 w2html
            cd $(htmldir) && ./w2html nlpp.w
```

◇

Fragment defined by 79bd, 80ab, 82a, 83d, 85c, 86b.
Fragment referenced in 78b.

Invoke:

⟨ *make targets* 86c ⟩ ≡
```
    htm : $(htmldir) $(htmltarget)
```

◇

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.

Create a script that performs the translation.

"w2html" 86d≡
```
    #!/bin/bash
    # w2html -- make a html file from a nuweb file
    # usage: w2html [filename]
    #  [filename]: Name of the nuweb source file.
    # 20170510 at 1359h: Generated by nuweb from a_nlpp.w
    echo "translate " $1 >w2html.log
    NUWEB=/home/paul/projecten/pipelines/nlpp/env/bin/nuweb
```
    ⟨ *filenames in w2html* 87a ⟩

    ⟨ *perform the task of w2html* 86e ⟩

◇

Uses: nuweb 84d.

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* 86e ⟩ ≡
    ⟨ *run the html processors until the aux file remains unchanged* 87b ⟩
    ⟨ *remove the copy of the aux file* 84c ⟩
◇

Fragment referenced in 86d.

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* 87a ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
```
◇

Fragment referenced in 86d.
Defines: auxfil 84b, 85a, 87b, nufil 84bd, 87c, oldaux 84bc, 85a, 87b, texfil 84bd, 87c, trunk 84bd, 87cd.
Uses: indexfil 84b, oldindexfil 84b.

⟨ *run the html processors until the aux file remains unchanged* 87b ⟩ ≡
```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 87c ⟩
done
⟨ run tex4ht 87d ⟩
```

◇

Fragment referenced in 86e.
Uses: auxfil 84b, 87a, oldaux 84b, 87a.

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

⟨ *run the html processors* 87c ⟩ ≡
```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
```
◇
Fragment referenced in 87b.
Uses: bibtex 84d, makeindex 84d, nufil 84b, 87a, texfil 84b, 87a, trunk 84b, 87a.

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* 87d ⟩ ≡
```
tex '\def\filename{{nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
```
◇
Fragment referenced in 87b.
Uses: bibtex 84d, makeindex 84d, trunk 84b, 87a.

### 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* 88a ⟩ ≡
```
      MKDIR = mkdir -p
```

      ⋄

Fragment defined by 78a, 79a, 81ab, 83c, 85b, 88a.
Fragment referenced in 78b.
Defines: MKDIR 88b.

⟨ *make targets* 88b ⟩ ≡
```
      DIRS = ⟨ directories to create 6, ... ⟩

      $(DIRS) :
              $(MKDIR) $@
```

      ⋄

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Defines: DIRS 88d.
Uses: MKDIR 88a.

⟨ *make scripts executable* 88c ⟩ ≡
```
      chmod -R 775  ../bin/*
      chmod -R 775  ../env/bin/*
```
      ⋄

Fragment defined by 26d, 37a, 88c.
Fragment referenced in 88d.

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* 88d ⟩ ≡
```
      sources : nlpp.w $(DIRS) $(NUWEB)
              $(NUWEB) nlpp.w
              ⟨ make scripts executable 26d, ... ⟩
```

      ⋄

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Uses: DIRS 88b.

The "install" target performs the complete installation.

⟨ *make targets* 89a ⟩ ≡
```
install : sources
        ../bin/install-modules
```

◇

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Defines: `install` 10c, 12ab, 16b, 17a, 18ac, 19ae, 26dj, 27a, 31d, 33c, 43a, 45d, 61a, 62c, 63f, 64ab, 66a, 68c, 70a, 78e, 89b.

## 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* 89b ⟩ ≡

```
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 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.
Defines: `testen` Never used, `testnl` Never used.
Uses: `install` 89a.

## 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* 89c ⟩ ≡
```
transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant
```

◇

Fragment defined by 78d, 82b, 83a, 86c, 88bd, 89abc.
Fragment referenced in 78b.

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

```
"../env/bin/transplant" 90≡
      #!/bin/bash
      LOGLEVEL=1
```
⟨ *set variables that point to the directory-structure* 7d, . . . ⟩
⟨ *set paths after transplantation* 18b ⟩
⟨ *re-install modules after the transplantation* 33d ⟩


      ◇




# 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

`"../bin/tok"` Defined by 41c.
`"../bin/topic"` Defined by 43b.
`"../bin/ukb"` Defined by 46d.
`"../bin/wikify"` Defined by 46a.
`"../bin/wsd"` Defined by 59c.
`"../env/bin/langdetect.py"` Defined by 29b.
`"../env/bin/progenv"` Defined by 8a, 13b, 48a.
`"../env/bin/transplant"` Defined by 90.
`"../nuweb/bin/w2pdf"` Defined by 83e.
`"Makefile"` Defined by 78b.
`"w2html"` Defined by 86d.

## C.2   Macro's

⟨ activate the install-to-project-repo utility 64b ⟩ Not referenced.
⟨ activate the python environment 17bde ⟩ Referenced in 15e, 22a.
⟨ all targets 78e ⟩ Referenced in 78c.
⟨ annotate 73 ⟩ Referenced in 74b, 75a.
⟨ annotate dutch document 72a ⟩ Not referenced.
⟨ annotate english document 72b ⟩ Referenced in 73.
⟨ begin conditional install 20c ⟩ Referenced in 13d, 22a, 23aj, 24aj, 25ahq, 26a.
⟨ check listener on host, port 37d ⟩ Referenced in 36c, 40a.
⟨ check this first 10g, 26e ⟩ Referenced in 22a.
⟨ check whether mercurial is present 27a ⟩ Referenced in 26e.
⟨ check whether program is present 26j ⟩ Referenced in 26e.
⟨ check/install the correct version of python 16a ⟩ Referenced in 15e.
⟨ clean up 13g, 15a, 31b, 79c ⟩ Referenced in 78d.
⟨ clone the heideltime wrapper 63d ⟩ Referenced in 63c.
⟨ compile nuweb 84a ⟩ Referenced in 83e.
⟨ compile the heideltime wrapper 64a ⟩ Referenced in 63c.
⟨ compile the nerc jar 57b ⟩ Referenced in 57a.
⟨ compile the topic-tool jar 43a ⟩ Referenced in 42a.
⟨ create a virtual environment for Python 16c ⟩ Referenced in 15e.
⟨ create javapython script 13a ⟩ Referenced in 22a.
⟨ default target 78c ⟩ Referenced in 78b.
⟨ directories to create 6, 7abc, 13c, 14bf, 17c, 83b ⟩ Referenced in 88b.
⟨ download everything 9c ⟩ Referenced in 22a.
⟨ download stuff 11ac, 14c, 19b, 34b, 40e, 42b, 60d ⟩ Referenced in 9c.
⟨ download svm models 59b ⟩ Referenced in 58e.
⟨ else conditional install 21a ⟩ Not referenced.
⟨ end conditional install 21b ⟩ Referenced in 13d, 22a, 23aj, 24aj, 25ahq, 26a.
⟨ expliciete make regels 79bd, 80ab, 82a, 83d, 85c, 86b ⟩ Referenced in 78b.
⟨ filenames in nuweb compile script 84b ⟩ Referenced in 83e.
⟨ filenames in w2html 87a ⟩ Referenced in 86d.
⟨ find a spotlightserver or exit 37b ⟩ Referenced in 46a, 61b.
⟨ function to run a module 71c ⟩ Referenced in 74b, 75a.
⟨ function to start EHU SRL server 49a ⟩ Referenced in 50a.
⟨ get a testfile or die 74a ⟩ Referenced in 74b.
⟨ get commandline-arguments 27b ⟩ Referenced in 36a.
⟨ get spotlight language parameters 37c ⟩ Not referenced.
⟨ get spotlight model ball 35b ⟩ Not referenced.
⟨ get the mor time-out parameter 44a ⟩ Referenced in 43e.
⟨ get the nerc models 57d ⟩ Referenced in 57a.
⟨ get the path to the module-script 29a ⟩ Referenced in 28b.
⟨ impliciete make regels 81c, 86a ⟩ Referenced in 78b.
⟨ install ActivePython 16b ⟩ Referenced in 16a.
⟨ install Alpino 30c ⟩ Referenced in 22a.
⟨ install boost 34a ⟩ Referenced in 22a.

⟨ install coreference-base 56c ⟩ Referenced in 25a.
⟨ install from github 10a ⟩ Referenced in 43d, 50f, 55c, 58e, 60c, 63d, 65a, 67d, 69c.
⟨ install from tarball 76a ⟩ Not referenced.
⟨ install Java 1.6 15c ⟩ Referenced in 22a.
⟨ install kafnafparserpy 18c ⟩ Referenced in 15e.
⟨ install libxml2 or libxslt 12b ⟩ Referenced in 12c.
⟨ install maven 14g ⟩ Referenced in 22a.
⟨ install naf2sem 68c ⟩ Not referenced.
⟨ install perl 19ef, 20a ⟩ Referenced in 22a.
⟨ install python packages 19a, 66a ⟩ Referenced in 15e.
⟨ install sematree 28a ⟩ Referenced in 22a.
⟨ install shared libs 12ac ⟩ Referenced in 22a.
⟨ install svm lib 59a ⟩ Referenced in 58e.
⟨ install SVMLight 41a ⟩ Referenced in 22a.
⟨ install the constituents parser 45a ⟩ Referenced in 23j.
⟨ install the dbpedia-ner module 67d ⟩ Referenced in 25q.
⟨ install the de-link module 61d ⟩ Referenced in 23j.
⟨ install the event-coreference module 67a ⟩ Referenced in 25h.
⟨ install the factuality modules 55c ⟩ Referenced in 24j.
⟨ install the FBK-causalrel module 54c ⟩ Referenced in 24j.
⟨ install the FBK-temprel module 53b ⟩ Referenced in 24j.
⟨ install the FBK-time module 51c ⟩ Referenced in 24j.
⟨ install the heideltime module 63c ⟩ Referenced in 25h.
⟨ install the ims-wsd module 47a ⟩ Referenced in 24a.
⟨ install the jex resources and libraries 42g ⟩ Referenced in 42a.
⟨ install the lu2synset converter 59e ⟩ Referenced in 25h.
⟨ install the morphosyntactic parser 43d ⟩ Referenced in 23a.
⟨ install the NERC module 57a ⟩ Referenced in 23j.
⟨ install the ontotagger repository 62c ⟩ Referenced in 25a.
⟨ install the opinion-miner 69c, 70abcd ⟩ Referenced in 26a.
⟨ install the pos tagger 44b ⟩ Referenced in 23a.
⟨ install the post-SRL module 66b ⟩ Referenced in 25q.
⟨ install the Spotlight server 35ac ⟩ Referenced in 22a.
⟨ install the srl module 65a ⟩ Referenced in 25h.
⟨ install the srl-dutch-nominals module 50f ⟩ Referenced in 24a.
⟨ install the srl-server module 49c ⟩ Referenced in 24a.
⟨ install the ticcutils utility 33a ⟩ Referenced in 22a, 33d.
⟨ install the timbl utility 33b ⟩ Referenced in 22a, 33d.
⟨ install the tokenizer 41b ⟩ Referenced in 23a.
⟨ install the topic analyser 42a ⟩ Referenced in 23a.
⟨ install the treetagger utility 31cde, 32abcd ⟩ Referenced in 22a.
⟨ install the UKB module 46c ⟩ Not referenced.
⟨ install the wikify module 45g ⟩ Referenced in 23j.
⟨ install the WSD module 58e ⟩ Referenced in 25a.
⟨ install the NED-reranker module 45d ⟩ Referenced in 23j.
⟨ install the NED module 60c ⟩ Referenced in 23j.
⟨ install VUA-pylib 40d ⟩ Referenced in 22a.
⟨ logmess 75d ⟩ Referenced in 9a, 10a, 76a.
⟨ make scripts executable 26d, 37a, 88c ⟩ Referenced in 88d.
⟨ make targets 78d, 82b, 83a, 86c, 88bd, 89abc ⟩ Referenced in 78b.
⟨ move module 8b ⟩ Referenced in 10a, 76a.
⟨ need to wget 11b ⟩ Referenced in 11c, 14c, 19b, 34b, 40e, 42b, 60d.
⟨ parameters in Makefile 78a, 79a, 81ab, 83c, 85b, 88a ⟩ Referenced in 78b.
⟨ perform the task of w2html 86e ⟩ Referenced in 86d.
⟨ put miscellaneous stuff in the heideltime module 63f ⟩ Referenced in 63c.
⟨ put spotlight jar in the Maven repository 61a ⟩ Referenced in 60c.
⟨ re-install modules after the transplantation 33d ⟩ Referenced in 90.
⟨ re-instate old module 9a ⟩ Referenced in 10a, 76a.

⟨ read the list of installed modules 20b ⟩ Referenced in 22a.
⟨ remove installed-variable 21c ⟩ Referenced in 15a.
⟨ remove old module 8c ⟩ Referenced in 10a, 76a.
⟨ remove outdated heideltime jars 64c ⟩ Referenced in 64b.
⟨ remove the copy of the aux file 84c ⟩ Referenced in 84a, 86e.
⟨ repair causalrel's run.sh.hadoop 54e ⟩ Not referenced.
⟨ repair FBK-*rel's run.sh.hadoop 53d ⟩ Referenced in 53b, 54c.
⟨ run dutch factuality 56b ⟩ Referenced in 55d.
⟨ run english factuality 56a ⟩ Referenced in 55d.
⟨ run in subshell when naflang is not known 30a ⟩ Referenced in 28b.
⟨ run only if language is English or Dutch 30b ⟩ Referenced in 28b.
⟨ run tex4ht 87d ⟩ Referenced in 87b.
⟨ run the html processors 87c ⟩ Referenced in 87b.
⟨ run the html processors until the aux file remains unchanged 87b ⟩ Referenced in 86e.
⟨ run the processors until the aux file remains unchanged 85a ⟩ Referenced in 84a.
⟨ run the three processors 84d ⟩ Referenced in 85a.
⟨ select language-dependent features 57c ⟩ Not referenced.
⟨ set alpinohome 31a ⟩ Referenced in 43e.
⟨ set default arguments for Spotlight 36b ⟩ Referenced in 36a.
⟨ set paths after transplantation 18b ⟩ Referenced in 90.
⟨ set up java 13d, 14a ⟩ Referenced in 22a.
⟨ set up Java 1.6 15d ⟩ Referenced in 47b.
⟨ set up python 15e ⟩ Referenced in 22a.
⟨ set variables for test/run script 75b ⟩ Referenced in 74b, 75a.
⟨ set variables that point to the directory-structure 7de, 10f, 14h ⟩ Referenced in 8a, 22a, 90.
⟨ start EHU SRL server 50e ⟩ Not referenced.
⟨ start EHU SRL server if it isn't running 48b ⟩ Referenced in 50a.
⟨ start of module-script 28b ⟩ Referenced in 41c, 43be, 44c, 45be, 46ad, 47b, 50c, 51a, 52a, 54a, 55ad, 56d, 58ac, 59c, 60a, 61b, 62adf, 63a, 64d, 65b, 66c, 67b, 68a, 69a, 71a.
⟨ start the Spotlight server on localhost 39ab ⟩ Referenced in 36c, 38a.
⟨ stop on error 53a ⟩ Referenced in 52a.
⟨ test whether spotlighthost runs 38e ⟩ Referenced in 38a.
⟨ test whether virtualenv is present on the host 17a ⟩ Referenced in 16c.
⟨ try to obtain a running spotlightserver 38a ⟩ Not referenced.
⟨ unpack ticcutils or timbl 33c ⟩ Referenced in 33ab.
⟨ update pip 18a ⟩ Referenced in 15e.
⟨ variables of install-modules 75c ⟩ Referenced in 22a.
⟨ wait for SRL server 49b ⟩ Referenced in 50c.
⟨ wait until the spotlight server is up or faulty 40a ⟩ Referenced in 39b.

## C.3 Variables

`activate`: <u>17b</u>, 18b.
`all`: 69a, <u>78c</u>.
`ALPINO_HOME`: <u>31a</u>.
`auxfil`: <u>84b</u>, 85a, <u>87a</u>, 87b.
`bibtex`: <u>84d</u>, 87cd.
`BIND`: <u>71c</u>, 75b.
`DIRS`: <u>88b</u>, 88d.
`fig2dev`: <u>81c</u>.
`FIGFILENAMES`: <u>81b</u>.
`FIGFILES`: <u>81a</u>, 81b.
`hg`: <u>27a</u>.
`indexfil`: <u>84b</u>, 85a, 87a.
`install`: 10c, 12ab, 16b, 17a, 18ac, 19ae, 26dj, 27a, 31d, 33c, 43a, 45d, 61a, 62c, 63f, 64ab, 66a, 68c, 70a, 78e, <u>89a</u>, 89b.
`LD_LIBRARY_PATH`: <u>17e</u>, 46d, 56a.
`lxml`: <u>19a</u>.