Standardised Dutch NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

27th November 2015
10:02 h.

Abstract

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

Contents

1 Introduction

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

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

Apart from describing the pipeline set-up, the document actually constructs the pipeline. Currently, the pipeline has been succesfully implemented on a specific supercomputer (Lisa, Surfsara, Amsterdam [3]) and on computers running Ubuntu and Centos.

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.

1.1 List of the modules to be installed

Table 1 lists the modules in the pipeline. The column *source* indicates the origin of the module.

Module	Section	Source	Commit	Script
Tokenizer	4.4.1	Github	56f83ce4b61680346f15e5d4e6de6293764f7383	`tok`
morphosyntactic parser	4.4.2	Github	c6cabea2cc37ac3098c5927f5ec5b180ac31246f	`mor`
NERC	4.4.4	Gith./snap	5cacac28fcaa6e91d5f2a4cc9b486b24ac163641	nerc
WSD	4.4.5	Gith./snap	2babeb40a81b3720274a0521ccc2a27c5eff28c9	wsd
Onto-tagger	4.4.8	snapshot		onto
Heideltime	4.4.10	Gith./snap.	da4604a7b33975e977017440cbc10f7d59917ddf	heideltime
SRL	4.4.11	Github	675d22d361289ede23df11dcdb17195f008c54bf	srl
SRL-POST	4.4.12	snapshot		postsrl
NED	4.4.7	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	4.4.3	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	4.4.13	snapshot		evcoref
Opinion miner	4.4.16	Github		opinimin
Framenet SRL	4.4.9	snapshot		fsrl
Dbpedia_ner	4.4.14	Github	ab1dcbd860f0ff29bc979f646dc382122a101fc2	dbpner

Table 1: *List of the modules to be installed. Column description:* ***directory:*** *Name of the subdirectory below subdirectory* `modules` *in which it is installed;* ***source:*** *From where the module has been obtained;* ***commit:*** *Commit-name or version-tag* ***script:*** *Script to be included in a pipeline.* ***Note:*** *The tokenizer module has been temporarily obtained from the snapshot, because the commit that we used has disappeared from the Github repository.*

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 1 this has been indicated as SNAPSHOT.

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

Module	Version	Section	Source
KafNafParserPy	Feb 1, 2015	3.3.4	Github
Alpino	20706	4.3.1	RUG
Ticcutils	0.7	4.3.3	ILK
Timbl	6.4.6	4.3.3	ILK
Treetagger	3.2	4.3.2	Uni. München
Spotlight server	0.7	4.3.4	Spotlight

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

1.2 File-structure of the pipeline

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

3. https://surfsara.nl/systems/lisa

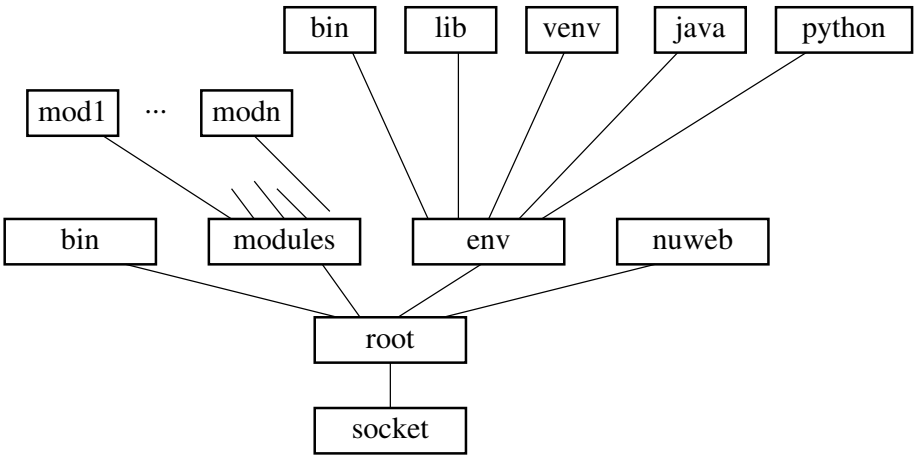

Figure 1: *Directory-structure of the pipeline (see text).*

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* 4a ⟩ ≡
```
../modules ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

⟨ *directories to create* 4b ⟩ ≡
```
../bin ../env/bin ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

⟨ *directories to create* 4c ⟩ ≡
```
../env/lib ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

⟨ *directories to create* 4d ⟩ ≡
```
../env/etc ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

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* 5a ⟩ ≡

```
if
  [ "$piperoot" == "" ]
then
  export piperoot=/home/huygen/projecten/pipelines/nlpp
fi
export pipesocket=${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 5ab.
Fragment referenced in 5c, 15b, 54.
Uses: `nuweb` 49b.

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

⟨ *set variables that point to the directory-structure* 5b ⟩ ≡

```
export PATH=$envbindir:$PATH
◇
```

Fragment defined by 5ab.
Fragment referenced in 5c, 15b, 54.
Defines: `PATH` 9e, 10b, 39a.

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

`"../env/bin/progenv"` 5c≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
export progenvset=0
◇
```

File defined by 5c, 8c.

2 How to obtain modules and other material

As illustrated in tables 1 and 2, 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.8).

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* 6a ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
◇
```
Fragment referenced in 7a, 14c, 40c.

⟨ *remove old module* 6b ⟩ ≡
```
rm -rf old.@1
◇
```
Fragment referenced in 7a, 14c, 40c.

⟨ *re-instate old module* 6c ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
⟨ logmess (6d $MESS ) 40b ⟩

◇
```
Fragment referenced in 7a, 14c, 40c.

2.3 Installation from Github

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

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

⟨ *install from github* 7a ⟩ ≡

```
cd $modulesdir
⟨ move module (7b $DIRN ) 6a ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (7c Cannot install current $MODNAM version ) 40b ⟩
  ⟨ re-instate old module (7d $DIRN ) 6c ⟩
else
  ⟨ remove old module (7e $DIRN ) 6b ⟩
  cd $modulesdir/$DIRN
  git checkout $GITC
fi

◇
```

Fragment referenced in 25ac, 28a, 29d, 32c, 34c, 37a.

2.4 Installation from the snapshot

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* 7f ⟩ ≡

```
if
  [ ! -e $pipesocket/nrkey ]
then
  echo "No key to connect to snapshot!"
  exit 1
fi
◇
```

Fragment defined by 7f, 16c.
Fragment referenced in 15b.

Use the following macro to download a resource if it is not already present in the “socket” directory. It turns out that sometimes there is a time-out for unknown reasons. In that case we will try it multiple times.

⟨ *get or have* 8a ⟩ ≡

```
counter=0
while
  [ ! -e $pipesocket/@1 ]
do
  cd $pipesocket
  scp -
i "/home/huygen/projecten/pipelines/nrkey" newsreader@kyoto.let.vu.nl:nlpp_resources/@1 .
  if
    [ $? -gt 0 ]
  then
    counter=$((counter+1))
    if
      [ $counter -gt 3 ]
    then
       echo "Cannot contact snapshot server"
       exit 1
    fi
  fi
done
```

◇

Fragment referenced in 9b, 11b, 17b, 20a, 21a, 27a, 28c, 29a, 30c, 33a, 35e, 36b, 37c, 38a.

3 Java and Python environment

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

The following 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* 8b ⟩ ≡

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

◇

Fragment referenced in 15b.

Cause the module scripts to read the javapython script.

"../env/bin/progenv" 8c≡

```
source $envbindir/javapython
```

◇

File defined by 5c, 8c.

3.1 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* 9a ⟩ ≡
```
../env/java ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

⟨ *set up java* 9b ⟩ ≡
```
⟨ get or have (9c server-jre-7u72-linux-x64.tar.gz ) 8a ⟩
cd $envdir/java
tar -xzf $pipesocket/server-jre-7u72-linux-x64.tar.gz
◇
```
Fragment defined by 9be.
Fragment referenced in 15b.

Remove the java-ball when cleaning up:

⟨ *clean up* 9d ⟩ ≡
```
rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz
◇
```
Fragment defined by 9d, 10c, 17f, 35g, 44a.
Fragment referenced in 43a.

⟨ *set up java* 9e ⟩ ≡
```
echo 'export JAVA_HOME=$envdir/java/jdk1.7.0_72' >> /home/huygen/projecten/pipelines/nlpp/env/bin/jav
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /home/huygen/projecten/pipelines/nlpp/env/bin/javapython
export JAVA_HOME=$envdir/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 9be.
Fragment referenced in 15b.
Uses: PATH 5b.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 9f ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

3.2 Maven

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

⟨ *directories to create* 9g ⟩ ≡
```
../env/apache-maven-3.0.5 ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

⟨ *install maven* 10a ⟩ ≡

```
cd $envdir
wget http://apache.rediris.es/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-
bin.tar.gz
tar -xzf apache-maven-3.0.5-bin.tar.gz
rm apache-maven-3.0.5-bin.tar.gz
◇
```

Fragment defined by 10ab.
Fragment referenced in 15b.

⟨ *install maven* 10b ⟩ ≡

```
export MAVEN_HOME=$envdir/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```

Fragment defined by 10ab.
Fragment referenced in 15b.
Uses: PATH 5b.

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

⟨ *clean up* 10c ⟩ ≡

```
rm -rf ../env/apache-maven-3.0.5
◇
```

Fragment defined by 9d, 10c, 17f, 35g, 44a.
Fragment referenced in 43a.

3.3 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* 10d ⟩ ≡

```
⟨ check/install the correct version of python 11a ⟩
⟨ create a virtual environment for Python 13b ⟩
⟨ activate the python environment 13d, ... ⟩
⟨ install kafnafparserpy 14c ⟩
⟨ install python packages 15a ⟩
◇
```

Fragment referenced in 15b.

⟨ *check/install the correct version of python* 11a ⟩ ≡

```
pythonok=‘python --
version 2>&1 | gawk ’{if(match($2, "2.7")) print "yes" ; else print "no" }’‘
if
  [ "$pythonok" == "no" ]
then
  ⟨ install ActivePython 11b ⟩
fi
◇
```

Fragment referenced in 10d.
Defines: `pythonok` Never used.
Uses: `print` 47a.

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* 11b ⟩ ≡

```
⟨ get or have (11c ActivePython-2.7.8.10-linux-x86_64.tar.gz ) 8a ⟩
pytinsdir=‘mktemp -d -t activepyt.XXXXXX‘
cd $pytinsdir
tar -xzf $pipesocket/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 pip virtualenv setuptools
◇
```

Fragment referenced in 11a.

3.3.1 Transplant ActivePython

Activepython produces scripts in `env/bin` that contain “shabangs” with absolute path. Furthermore, activePython seems to have an implicit pythonpath with an absolute path. So, when transplanting the directorytree to another location we have to solve these two problems.

While doing this, we also modify the scripts in the Python Virtenv binary directory (see 3.3.2).

Modify the scripts as follows:

1. Create a temporary directory.
2. Generate an AWK script that replaces the shabang line with a correct one.
3. Generate a script that moves a script from `env/bin` to the temporary directory and then applies the AWK script.
4. Apply the generated script on the scripts in `env/bin`.

⟨ *set paths after transplantation* 12a ⟩ ≡

```
transdir=‘mktemp -d -t trans.XXXXXX‘
cd $transdir
⟨ write script tran 12d ⟩
⟨ write script chasbang.awk 12e ⟩
⟨ apply script tran on the scripts in (12b $envbindir ) 12f ⟩
⟨ apply script tran on the scripts in (12c $envdir/venv/bin ) 12f ⟩
cd $projroot
rm -rf $transdir
◇
```

Fragment defined by 12a, 13a, 14b.
Fragment referenced in 54.

⟨ *write script tran* 12d ⟩ ≡

```
cat <<EOF >tran
workfil=$1
mv $workfil ./wor
gawk -f chasbang.awk ./wor >$workfil
EOF
chmod 775 ./tran
◇
```

Fragment referenced in 12a.

⟨ *write script chasbang.awk* 12e ⟩ ≡

```
cat <<EOF >chasbang.awk
#!/usr/bin/gawk -f
BEGIN { shabang="#!$envbindir/python"}

/^\#\!.*python.*/ { print shabang
                    next
                  }
{print}
EOF
◇
```

Fragment referenced in 12a.
Uses: print 47a.

The following looks complicated. The `find` command applies the `file` command on the files in the `env/bin` directory. The grep command filters out the names of the files that are scripts. it produces a filename, followed by a colon, followed by a description of the type of the file. The `gawk` command prints the filenames only and the `xargs` command applies the `tran` script on the file.

⟨ *apply script tran on the scripts in* 12f ⟩ ≡

```
find @1 -type f -exec file {} + | grep script | gawk ’{print $1}’ FS=’:’ | xargs -
iaap ./tran aap
◇
```

Fragment referenced in 12a.
Uses: print 47a.

Add `env/lib/python2.7` to the `PYTHONPATH` variable.

⟨ *set paths after transplantation* 13a ⟩ ≡
```
echo export PYTHONPATH=\$envdir/lib/python2.7:\$PYTHONPATH >> $envbindir/javapython
export PYTHONPATH=\$envdir/lib/python2.7:\$PYTHONPATH
◇
```
Fragment defined by 12a, 13a, 14b.
Fragment referenced in 54.
Uses: PYTHONPATH 14a.

3.3.2 Virtual environment

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

⟨ *create a virtual environment for Python* 13b ⟩ ≡
```
⟨ test whether virtualenv is present on the host 13c ⟩
cd $envdir
virtualenv venv
◇
```
Fragment referenced in 10d.
Uses: virtualenv 13c.

⟨ *test whether virtualenv is present on the host* 13c ⟩ ≡
```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
◇
```
Fragment referenced in 13b.
Defines: virtualenv 11b, 13b.

⟨ *activate the python environment* 13d ⟩ ≡
```
source $envdir/venv/bin/activate
echo 'source $en-
vdir/venv/bin/activate' >> /home/huygen/projecten/pipelines/nlpp/env/bin/javapython
◇
```
Fragment defined by 13d, 14a.
Fragment referenced in 10d.
Defines: activate 14b.

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

⟨ *directories to create* 13e ⟩ ≡
```
../env/python ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.

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

⟨ *activate the python environment* 14a ⟩ ≡
```
echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /home/huygen/projecten/pipelines/nlpp/env/bin/javapyt
export PYTHONPATH=$envdir/python:$PYTHONPATH
◇
```
Fragment defined by 13d, 14a.
Fragment referenced in 10d.
Defines: PYTHONPATH 13a.

3.3.3 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* 14b ⟩ ≡
```
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 defined by 12a, 13a, 14b.
Fragment referenced in 54.
Uses: activate 13d, print 47a.

3.3.4 KafNafParserPy

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

⟨ *install kafnafparserpy* 14c ⟩ ≡
```
cd $envdir/python
DIRN=KafNafParserPy
⟨ move module (14d $DIRN ) 6a ⟩
git clone https://github.com/cltl/KafNafParserPy.git
if
  [ $? -gt 0 ]
then
  ⟨ logmess (14e Cannot install current $DIRN version ) 40b ⟩
  ⟨ re-instate old module (14f $DIRN ) 6c ⟩
else
  ⟨ remove old module (14g $DIRN ) 6b ⟩
fi
◇
```
Fragment referenced in 10d.

3.3.5 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* 15a ⟩ ≡
```
pip install lxml
pip install pyyaml
```
◇

Fragment referenced in 10d.
Defines: `lxml` Never used, `pyyaml` Never used.

4 Installation of the modules

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

4.1 The installation script

The installation is performed by script `install-modules`. The first part of the script installs the utilities:

`"../bin/install-modules"` 15b≡
```
#!/bin/bash
echo Set up environment
⟨ set variables that point to the directory-structure 5a, ... ⟩
⟨ variables of install-modules 40a ⟩
⟨ check this first 7f, ... ⟩
⟨ create javapython script 8b ⟩
echo ... Java
⟨ set up java 9b, ... ⟩
⟨ install maven 10a, ... ⟩
echo ... Python
⟨ set up python 10d ⟩
echo ... Alpino
⟨ install Alpino 17b ⟩
echo ... Spotlight
⟨ install the Spotlight server 21a, ... ⟩
echo ... Treetagger
⟨ install the treetagger utility 18a, ... ⟩
echo ... Ticcutils and Timbl
⟨ install the ticcutils utility 19c ⟩
⟨ install the timbl utility 19d ⟩
echo ... VUA-pylib, SVMlight, CRFsuite
⟨ install VUA-pylib 23a ⟩
⟨ install SVMLight 23b ⟩
⟨ install CRFsuite 24a ⟩

◇
```

File defined by 15b, 16a.

Next, install the modules:

```
"../bin/install-modules" 16a≡
        echo Install modules
        echo ... Tokenizer
        ⟨ install the tokenizer 24b ⟩
        echo ... Morphosyntactic parser
        ⟨ install the morphosyntactic parser 25a ⟩
        echo ... NERC
        ⟨ install the NERC module 26b ⟩
        echo ... Coreference base
        ⟨ install coreference-base 25c ⟩
        echo ... WSD
        ⟨ install the WSD module 28a ⟩
        echo ... Ontotagger
        ⟨ install the onto module 30c ⟩
        echo ... Heideltime
        ⟨ install the heideltime module 32b ⟩
        echo ... SRL
        ⟨ install the srl module 34c ⟩
        echo ... NED
        ⟨ install the NED module 29d ⟩
        echo ... Event-coreference
        ⟨ install the event-coreference module 36b ⟩
        echo ... lu2synset
        ⟨ install the lu2synset converter 29a ⟩
        echo ... dbpedia-ner
        ⟨ install the dbpedia-ner module 37a ⟩
        echo ... nominal event
        ⟨ install the nomevent module 37c ⟩
        ⟨ install the post-SRL module 35e ⟩
        ⟨ install the opinion-miner 38a, ... ⟩

        echo Final
        ◇
```

File defined by 15b, 16a.

```
⟨ make scripts executable 16b ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```

Fragment defined by 16b, 53b.
Fragment referenced in 53c.

4.2 Check availability of resources

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

```
⟨ check this first 16c ⟩ ≡
        ⟨ check whether mercurial is present 17a ⟩
        ◇
```

Fragment defined by 7f, 16c.
Fragment referenced in 15b.

⟨ *check whether mercurial is present* 17a ⟩ ≡
```
which hg
if
  [ $? -ne 0 ]
then
  echo Please install Mercurial.
  exit 1
fi
```
◇

Fragment referenced in 16c.
Defines: `hg` 25c.

4.3 Install utilities and resources

4.3.1 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* 17b ⟩ ≡
```
⟨ get or have (17c Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
⟨ logmess (17d Installed Alpino ) 40b ⟩
```
◇

Fragment referenced in 15b.

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* 17e ⟩ ≡
```
export ALPINO_HOME=$modulesdir/Alpino
```
◇

Fragment referenced in 25b.
Defines: `ALPINO_HOME` Never used.

Remove the tarball when cleaning up:

⟨ *clean up* 17f ⟩ ≡
```
rm -rf $pipesocket/Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
```
◇

Fragment defined by 9d, 10c, 17f, 35g, 44a.
Fragment referenced in 43a.

4.3.2 Treetagger

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

1. Download and unpack the Treetagger tarball. This generates the subdirectories `bin`, `cmd` and `doc`

2. Download and unpack the tagger-scripts tarball

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

⟨ *install the treetagger utility* 18a ⟩ ≡

```
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 18abcde, 19ab.
Fragment referenced in 15b.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 18b ⟩ ≡

```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
◇
```

Fragment defined by 18abcde, 19ab.
Fragment referenced in 15b.

Parametersets:

⟨ *install the treetagger utility* 18c ⟩ ≡

```
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 18abcde, 19ab.
Fragment referenced in 15b.

Download everything in the target directory:

⟨ *install the treetagger utility* 18d ⟩ ≡

```
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 18abcde, 19ab.
Fragment referenced in 15b.

Run the install-script:

⟨ *install the treetagger utility* 18e ⟩ ≡

```
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```

Fragment defined by 18abcde, 19ab.
Fragment referenced in 15b.

Make the treetagger utilities available for everybody.

⟨ *install the treetagger utility* 19a ⟩ ≡
```
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 18abcde, 19ab.
Fragment referenced in 15b.

Remove the tarballs:

⟨ *install the treetagger utility* 19b ⟩ ≡
```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```
Fragment defined by 18abcde, 19ab.
Fragment referenced in 15b.

4.3.3 Timbl and Ticcutils

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

1. Download the tarball in a temporary directory.
2. Unpack the tarball.
3. cd to the unpacked directory and perform `./configure`, `make` and `make install`. Note the argument that causes the files to be installed in the `lib` and the `bin` sub-directories of the `env` directory.

⟨ *install the ticcutils utility* 19c ⟩ ≡
```
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 20a ⟩
◇
```
Fragment referenced in 15b, 20c.

⟨ *install the timbl utility* 19d ⟩ ≡
```
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
⟨ unpack ticcutils or timbl 20a ⟩
◇
```
Fragment referenced in 15b, 20c.

⟨ *unpack ticcutils or timbl* 20a ⟩ ≡
⟨ *get or have* (20b $TARB) 8a ⟩
```
SUCCES=0
ticbeldir=‘mktemp -t -d tickbel.XXXXXX‘
cd $ticbeldir
tar -xzf $pipesocket/$TARB
cd $DIR
./configure --prefix=$envdir
make
make install
cd $piperoot
rm -rf $ticbeldir
```
◇
Fragment referenced in 19cd.

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

⟨ *re-install modules after the transplantation* 20c ⟩ ≡
⟨ *install the ticcutils utility* 19c ⟩
⟨ *install the timbl utility* 19d ⟩
◇
Fragment referenced in 54.

4.3.4 Spotlight

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

> The NED module works for English, Spanish, Dutch and Italian. The module returns multiple candidates and correspondences for all the languages. If you want to integrate it in your Dutch or Italian pipeline, you will need:
>
> 1. The jar file with the dbpedia-spotlight server. You need the version that Aitor developed in order to correctly use the ”candidates” option. You can copy it from the English VM. The jar file name is `dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar`
> 2. The Dutch/Italian model for the dbpedia-spotlight. You can download them from: http://spotlight.sztaki.hu/downloads/
> 3. The jar file with the NED module: `ixa-pipe-ned-1.0.jar`. You can copy it from the English VM too.
> 4. The file: `wikipedia-db.v1.tar.gz`. You can download it from: http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz. This file contains the required information to do the mappings between the wikipedia-entries. The zip file contains three files: wikipedia-db, wikipedia-db.p and wikipedia-db.t
>
> To start the dbpedia server: Italian server:
> ```
> java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar \
>     it http://localhost:2050/rest
> ```
>
> Dutch server:
> ```
> java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://local
> ```
>
> We set 8Gb for the English server, but the Italian and Dutch Spotlight will require less memory.

So, let us do that:

⟨ *install the Spotlight server* 21a ⟩ ≡
```
⟨ get or have (21b spotlightnl.tgz ) 8a ⟩
cd $envdir
tar -xzf $pipesocket/spotlightnl.tgz
cd $envdir/spotlight
wget http://spotlight.sztaki.hu/downloads/nl.tar.gz
tar -xzf nl.tar.gz
rm nl.tar.gz
◇
```
Fragment defined by 21ac.
Fragment referenced in 15b.

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

⟨ *install the Spotlight server* 21c ⟩ ≡
```
cd $envdir/spotlight
wget http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz
tar -xzf wikipedia-db.v1.tar.gz
rm  wikipedia-db.v1.tar.gz
◇
```
Fragment defined by 21ac.
Fragment referenced in 15b.

Script `bin/start-spotlight` starts spotlight if it is not already running. It does the following:

1. If variable `spotlighthost` exists, it checks whether Spotlight is already running on that host.
2. If Spotlight does not run on that host or if If variable `spotlighthost` does not exist, it sets variable `spotlighthost` to localhost and then checks whether Spotlight runs on localhost.
3. If Spotlight has not yet been found, install spotlight on localhost.
4. If a running spotlight has been found, set variable `spotlightrunning` to 0.

"../bin/start-spotlight" 21d≡
```
# NOTE: This script ought to be sourced.
# Afterwards, on success, the following variables exist:
# > spotlighthost
# > spotlightrunning
if
  [ ! $spotlightrunning ]
then
  [ $spotlighthost ] || export spotlighthost=130.37.53.38
  ⟨ try to obtain a running spotlightserver 22a ⟩
fi
◇
```

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* 22a ⟩ ≡
```
⟨ test whether spotlighthost runs (22b $spotlighthost ) 22e ⟩
if
  [ ! $spotlightrunning ]
then
  if
    [ "$spotlighthost" != "localhost" ]
  then
    export spotlighthost=localhost
    ⟨ test whether spotlighthost runs (22c $spotlighthost ) 22e ⟩
  fi
fi
if
  [ ! $spotlightrunning ]
then
  ⟨ start the Spotlight server on localhost 22f ⟩
  ⟨ test whether spotlighthost runs (22d $spotlighthost ) 22e ⟩
fi
if
  [ ! $spotlightrunning ]
then
  echo "Cannot start spotlight"
  exit 4
fi
◇
```
Fragment referenced in 21d.

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* 22e ⟩ ≡
```
exec 6<>/dev/tcp/@1/2060
if
  [ $? -eq 0 ]
then
  export spotlightrunning=0
else
  spotlightrunning=
fi
exec 6<&-
exec 6>&-
◇
```
Fragment referenced in 22a.

⟨ *start the Spotlight server on localhost* 22f ⟩ ≡
```
[ $progenvset ] || source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
cd /home/huygen/projecten/pipelines/nlpp/env/spotlight
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-
candidates.jar nl http://localhost:2060/rest  &
sleep 60
◇
```
Fragment referenced in 22a.

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.

4.3.5 VUA-pylib

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

⟨ *install VUA-pylib* 23a ⟩ ≡
```
cd $envdir/python
git clone https://github.com/cltl/VUA_pylib.git
◇
```
Fragment referenced in 15b.

4.3.6 SVMLight

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

Installation goes like this:

⟨ *install SVMLight* 23b ⟩ ≡
```
tempdir=‘mktemp -d -t SVMlight.XXXXXX‘
cd $tempdir
wget http://download.joachims.org/svm_light/current/svm_light.tar.gz
tar -xzf svm_light.tar.gz
make all
cp svm_classify /home/huygen/projecten/pipelines/nlpp/env/bin/
cp svm_learn /home/huygen/projecten/pipelines/nlpp/env/bin/
cd /home/huygen/projecten/pipelines/nlpp
rm -rf $tempdir
◇
```
Fragment referenced in 15b.
Uses: `all` 42c.

4.3.7 CRFsuite

CRFsuite is an implementation of Conditional Random Fields (CRF).

⟨ *install liblblbfgs* 23c ⟩ ≡
```
tempdir=‘mktemp -d -t liblblbfgs.XXXXXX‘
cd $tempdir
git clone https://github.com/chokkan/liblbfgs.git
cd lblblfgs
./autogen.sh
./configure --prefix=$envdir
make
make install
cd /home/huygen/projecten/pipelines/nlpp
rm -rf $tempdir
◇
```
Fragment referenced in 24a.

⟨ *install CRFsuite* 24a ⟩ ≡
⟨ *install liblblbfgs* 23c ⟩
```
tempdir=‘mktemp -d -t crfsuite.XXXXXX‘
cd $tempdir
wget https://github.com/downloads/chokkan/crfsuite/crfsuite-0.12.tar.gz
tar -xzf crfsuite-0.12.tar.gz
cd crfsuite-0.12
./configure --prefix=$envdir
make
make install
cd /home/huygen/projecten/pipelines/nlpp
rm -rf $tempdir
```
◇

Fragment referenced in 15b.

4.4 Install modules

4.4.1 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* 24b ⟩ ≡
```
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 16a.

Script The script runs the tokenizerscript.

"../bin/tok" 24c≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
java -Xmx1000m  -jar $JARFILE tok -l nl --inputkaf
```
◇

4.4.2 Morphosyntactic parser

Module

⟨ *install the morphosyntactic parser* 25a ⟩ ≡
```
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=c6cabea2cc37ac3098c5927f5ec5b180ac31246f
⟨ install from github 7a ⟩
cd $modulesdir/morphosyntactic_parser_nl
git checkout c6cabea2cc37ac3098c5927f5ec5b180ac31246f
◇
```
Fragment referenced in 16a.

Script

"../bin/mor" 25b≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
MODDIR=$modulesdir/morphosyntactic_parser_nl
⟨ set alpinohome 17e ⟩
cat | python $MODDIR/core/morph_syn_parser.py
◇
```

4.4.3 Nominal coreference-base

Get this thing from Github (https://github.com/opener-project/coreference-base/) and apply the instruction of https://github.com/opener-project/coreference-base/blob/master/core/README.md. We implement it, but it does not work yet, because it is too picky on the structure of the NAF format.

Module

⟨ *install coreference-base* 25c ⟩ ≡
```
MODNAM=coreference-base
DIRN=coreference-base
GITU=https://github.com/opener-project/coreference-base.git
GITC=bfa5aec0fa498e57fe14dd4d2c51365dd09a0757
⟨ install from github 7a ⟩
pip install --upgrade  hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
pip install --upgrade  networkx
◇
```
Fragment referenced in 16a.
Uses: hg 17a.

Script

```
"../bin/coreference-base" 26a≡
    #!/bin/bash
    source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
    cd $modulesdir/coreference-base/core
    cat | python -m corefgraph.process.file --language nl --singleton --sieves NO
    ◇
```

4.4.4 Named entity recognition (NERC)

Module The Nerc program can be installed from Github (https://github.com/ixa-ehu/ixa-pipe-nerc). However, the model that is needed is not publicly available. Therefore, models have been put in the snapshot-tarball.

```
⟨ install the NERC module 26b ⟩ ≡
    ⟨ compile the nerc jar 26c ⟩
    ⟨ get the nerc models 27a ⟩

    ◇
```

Fragment referenced in 16a.

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 26c ⟩ ≡
    TEMPDIR=‘mktemp -d -t nerc.XXXXXX‘
    cd $TEMPDIR
    git clone https://github.com/ixa-ehu/ixa-pipe-nerc
    cd ixa-pipe-nerc/
    git checkout 5cacac28fcaa6e91d5f2a4cc9b486b24ac163641
    mvn clean package
    mv target/ixa-pipe-nerc-1.5.2.jar $jarsdir/
    cd $nuwebdir
    rm -rf $TEMPDIR
    ◇
```

Fragment referenced in 26b.

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

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

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

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

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

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

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

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

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

⟨ *get the nerc models* 27a ⟩ ≡
⟨ *get or have* (27b `nerc-models-nl.tgz`) 8a ⟩
```
mkdir -p $modulesdir/nerc-models-nl
cd $modulesdir/nerc-models-nl
tar -xzf $pipesocket/nerc-models-nl.tgz
chmod -R 775 $modulesdir/nerc-models-nl
```
◇
Fragment referenced in 26b.

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

`"../bin/nerc_conll02"` 27c≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
MODDIR=$modulesdir/nerc-models-nl
JAR=$jarsdir/ixa-pipe-nerc-1.5.2.jar
MODEL=nl-clusters-conll02.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nl/$MODEL
```
◇

4.4.5 Wordsense-disambiguation

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

Module

⟨ *install the WSD module* 28a ⟩ ≡

```
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=2babeb40a81b3720274a0521ccc2a27c5eff28c9
⟨ install from github 7a ⟩
cd $modulesdir/svm_wsd
⟨ install svm lib 28b ⟩
⟨ download svm models 28c ⟩


◇
```

Fragment referenced in 16a.

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

⟨ *install svm lib* 28b ⟩ ≡

```
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 28a.

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

⟨ *download svm models* 28c ⟩ ≡

```
⟨ get or have (28d svm_wsd.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/svm_wsd.tgz


◇
```

Fragment referenced in 28a.

Script

"../bin/wsd" 28e≡

```
#!/bin/bash
# WSD -- wrapper for word-sense disambiguation
# 8 Jan 2014 Ruben Izquierdo
# 16 sep 2014 Paul Huygen
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
WSDDIR=$modulesdir/svm_wsd
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $WSDDIR/$WSDSCRIPT --naf
◇
```

4.4.6 Lexical-unit converter

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

⟨ *install the lu2synset converter* 29a ⟩ ≡
```
⟨ get or have (29b lu2synset.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/lu2synset.tgz
◇
```
Fragment referenced in 16a.

Script

"../bin/lu2synset" 29c≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
JAVALIBDIR=$modulesdir/lexicalunitconvertor/lib
RESOURCESDIR=$modulesdir/lexicalunitconvertor/resources
JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
java -Xmx812m -
cp  $JAVALIBDIR/$JARFILE vu.wntools.util.NafLexicalUnitToSynsetReferences \
   --wn-lmf "$RESOURCESDIR/cornetto2.1.lmf.xml" --format naf
◇
```

4.4.7 NED

The NED module is rather picky about the structure of the NAF file. In any case, it does not accept a file that has been produced by the ontotagger. Hence, in a pipeline 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* 29d ⟩ ≡
```
⟨ put spotlight jar in the Maven repository 30a ⟩
MODNAM=ned
DIRN=ixa-pipe-ned
GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
⟨ install from github 7a ⟩
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 16a.

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

⟨ *put spotlight jar in the Maven repository* 30a ⟩ ≡

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

cd $PROJROOT
rm -rf $tempdir
◇
```

Fragment referenced in 29d.

Script

"../bin/ned" 30b≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
JARDIR=$jarsdir
[ $spotlightrunning ] || source /home/huygen/projecten/pipelines/nlpp/bin/start-
spotlight
cat | java -Xmx1000m -jar $jarsdir/ixa-pipe-ned-1.1.1.jar -
H http://$spotlighthost -p 2060 -e candidates -i $envdir/spotlight/wikipedia-db -
n nlEn
◇
```

4.4.8 Ontotagger

We do not yet have a source-repository of the Ontotagger module. Therefore, install from a snapshot (`20150724_vua-ontotagger-v1.0.tar.gz`).

Module

⟨ *install the onto module* 30c ⟩ ≡

```
⟨ get or have (30d 20150724_vua-ontotagger-v1.0.tar.gz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/20150724_vua-ontotagger-v1.0.tar.gz
rm $pipesocket/20150724_vua-ontotagger-v1.0.tar.gz
chmod -R o+r $modulesdir/vua-ontotagger-v1.0
◇
```

Fragment referenced in 16a.

Script

"../bin/onto" 31≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
ONTODIR=$modulesdir/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=‘mktemp -t stap6.XXXXXX‘
cat >$TMPFIL

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

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

◇
```

4.4.9 Framenet SRL

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

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

"../bin/framesrl" 32a≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ONTODIR=$modulesdir/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=‘mktemp -t framesrl.XXXXXX‘
cat >$TMPFIL

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

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

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

◇
```

Uses: print 47a.

4.4.10 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* 32b ⟩ ≡

```
moduledir=/home/huygen/projecten/pipelines/nlpp/modules/ixa-pipe-time
⟨ clone the heideltime wrapper 32c ⟩
⟨ put Antske’s material in the heideltime wrapper 33a ⟩
⟨ compile the heideltime wrapper 33c ⟩
◇
```

Fragment referenced in 16a.

⟨ *clone the heideltime wrapper* 32c ⟩ ≡

```
MODNAM=heideltime
DIRN=ixa-pipe-time
GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
GITC=da4604a7b33975e977017440cbc10f7d59917ddf
⟨ install from github (32d ixa-pipe-time ) 7a ⟩
mkdir $moduledir/lib
◇
```

Fragment referenced in 32b.

In the wrapper we need the following extra material:

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

The extra material has been provided by Antske Fokkens.

⟨ *put Antske's material in the heideltime wrapper* 33a ⟩ ≡
```
⟨ get or have (33b 20151123_antske_heideltime_stuff.tgz ) 8a ⟩
cd $modulesdir/$DIRN
tar -xzf /home/huygen/projecten/pipelines/20151123_antske_heideltime_stuff.tgz
mv antske_heideltime_stuff/de.unihd.dbs.heideltime.standalone.jar lib/
mv antske_heideltime_stuff/config.props .
mv antske_heideltime_stuff/alpino-to-treetagger.csv .
rm -rf antske_heideltime_stuff
◇
```
Fragment referenced in 32b.

Compile the Heideltime wrapper according to the instruction on Github.

⟨ *compile the heideltime wrapper* 33c ⟩ ≡
```
⟨ get jvntextpro-2.0.jar 33d ⟩
⟨ activate the install-to-project-repo utility 33e ⟩
cd /home/huygen/projecten/pipelines/nlpp/modules/$DIRN
mvn clean install
◇
```
Fragment referenced in 32b.

⟨ *get jvntextpro-2.0.jar* 33d ⟩ ≡
```
cd  /home/huygen/projecten/pipelines/nlpp/modules/$DIRN/lib
wget http://ixa2.si.ehu.es/%7Ejibalari/jvntextpro-2.0.jar
◇
```
Fragment referenced in 33c.

Scipt `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* 33e ⟩ ≡
```
⟨ remove outdated heideltime jars 34a ⟩
cd /home/huygen/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 referenced in 33c.

⟨ *remove outdated heideltime jars* 34a ⟩ ≡

```
rm -rf /home/huygen/projecten/pipelines/nlpp/modules/$DIRN/repo
mkdir -p /home/huygen/projecten/pipelines/nlpp/modules/$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 33e.

Script

"../bin/heideltime" 34b≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
HEIDELDIR=$modulesdir/ixa-pipe-time
cd $HEIDELDIR
iconv -t utf-8//IGNORE | java -jar target/ixa.pipe.time.jar -m alpino-to-
treetagger.csv -c config.props
◇
```

4.4.11 Semantic Role labelling

Module

⟨ *install the srl module* 34c ⟩ ≡

```
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=675d22d361289ede23df11dcdb17195f008c54bf
⟨ install from github 7a ⟩
◇
```

Fragment referenced in 16a.

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" 34d≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
ROOT=$piperoot
SRLDIR=$modulesdir/vua-srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $SRLDIR
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
◇
```

File defined by 34d, 35abcd.

Create a feature-vector.

```
"../bin/srl" 35a≡
    cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
    ◇
```

File defined by 34d, 35abcd.

Run the trained model on the feature-vector.

```
"../bin/srl" 35b≡
    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 34d, 35abcd.

Insert the SRL values into the NAF file.

```
"../bin/srl" 35c≡
    python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
    ◇
```

File defined by 34d, 35abcd.

Clean up.

```
"../bin/srl" 35d≡
    rm -rf $TEMPDIR
    ◇
```

File defined by 34d, 35abcd.

4.4.12 SRL postprocessing

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

Module Find the (Python) module in the snapshot and unpack it.

```
⟨ install the post-SRL module 35e ⟩ ≡
    ⟨ get or have (35f 20150706vua-srl-dutch-additional-roles.tgz ) 8a ⟩
    cd $modulesdir
    tar -xzf $pipesocket/20150706vua-srl-dutch-additional-roles.tgz
    ◇
```

Fragment referenced in 16a.

```
⟨ clean up 35g ⟩ ≡
    rm -rf $pipesocket/20150706vua-srl-dutch-additional-roles.tgz
    ◇
```

Fragment defined by 9d, 10c, 17f, 35g, 44a.
Fragment referenced in 43a.

Script

"../bin/postsrl" 36a≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
MODDIR=$modulesdir/vua-srl-dutch-additional-roles
cat | python $MODDIR/vua-srl-dutch-additional-roles.py
◇
```

4.4.13 Event coreference

Module Install the module from the snapshot.

⟨ *install the event-coreference module* 36b ⟩ ≡
```
⟨ get or have (36c 20150702-vua-eventcoreference_v2.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/20150702-vua-eventcoreference_v2.tgz
cd vua-eventcoreference_v2
cp lib/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar $jarsdir
◇
```
Fragment referenced in 16a.

Script

"../bin/evcoref" 36d≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
MODROOT=$modulesdir/vua-eventcoreference_v2
RESOURCESDIR=$MODROOT/resources
JARFILE=$jarsdir/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar

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


java -Xmx812m -cp $JARFILE $JAVAMODULE  $JAVAOPTIONS

◇
```

4.4.14 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* 37a ⟩ ≡
```
MODNAM=dbpedia_ner
DIRN=dbpedia_ner
GITU=https://github.com/PaulHuygen/dbpedia_ner.git
GITC=ab1dcbd860f0ff29bc979f646dc382122a101fc2
⟨ install from github 7a ⟩
◇
```
Fragment referenced in 16a.

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" 37b≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
[ $spotlightrunning ] || source ../bin/start-spotlight

MODDIR=$modulesdir/dbpedia_ner
cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
url http://$spotlighthost:2060/rest/candidates
◇
```

4.4.15 Nominal events

The module "postprocessing-nl" adds nominal events to the srl annotations. It has been obtained directly from the author (Piek Vossen). It is not yet available in a public repo. Probably in future versions the jar from the ontotagger module can be used for this module.

Module

⟨ *install the nomevent module* 37c ⟩ ≡
```
⟨ get or have (37d vua-postprocess-nl.zip ) 8a ⟩
cd $modulesdir
unzip -q $pipesocket/vua-postprocess-nl.zip
◇
```
Fragment referenced in 16a.

Script

"../bin/nomevent" 37e≡
```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
MODDIR=$modulesdir/vua-postprocess-nl
LIBDIR=$MODDIR/lib
RESOURCESDIR=$MODDIR/resources

JAR=$LIBDIR/ontotagger-1.0-jar-with-dependencies.jar
JAVAMODULE=eu.kyotoproject.main.NominalEventCoreference
cat | iconv -f ISO8859-1 -t UTF-8 | java -Xmx812m -cp $JAR $JAVAMODULE --framenet-
lu $RESOURCESDIR/nl-luIndex.xml
◇
```

4.4.16 Opinion miner

To run the opinion-miner, the following things are needed:

- SVMlight
- crfsuite
- vua-pylib

Module The module can be cloned from Github. However, currently there are problems with the Github installation. Therefore we borrow the opinion miner from the English NWR pipeline.

⟨ *install the opinion-miner* 38a ⟩ ≡
```
⟨ get or have (38b 20151012VUA-opinion-miner.tgz ) 8a ⟩
cd /home/huygen/projecten/pipelines/nlpp/modules
tar -xzf /home/huygen/projecten/pipelines/20151012VUA-opinion-miner.tgz
◇
```
Fragment defined by 38ad.
Fragment referenced in 16a.

The opinion-miner needs a configuration file that is located in the directory where the model-data resides. In this pipeline we will use model-data derived from news-articles. An alternative model, derived from hotel evaluations can also be used. Put the configuration file in the `etc` subdir and copy it to its proper location during the installation of the opinion-miner.

"../env/etc/opini_nl.cfg" 38c≡
```
[general]
output_folder = /home/huygen/projecten/pipelines/nlpp/modules/VUA-opinion-
miner/final_models/nl/news_cfg1

[crfsuite]
path_to_binary = /home/huygen/projecten/pipelines/nlpp/env/bin/crfsuite

[svmlight]
path_to_binary_learn = /home/huygen/projecten/pipelines/nlpp/env/bin/svm_learn
path_to_binary_classify = /home/huygen/projecten/pipelines/nlpp/env/bin/svm_classify
◇
```

⟨ *install the opinion-miner* 38d ⟩ ≡
```
cd VUA-opinion-miner
cp /home/huygen/projecten/pipelines/nlpp/env/etc/opini_nl.cfg $modulesdir/VUA-
opinion-miner/final_models/nl/news_cfg1/config.cfg
◇
```
Fragment defined by 38ad.
Fragment referenced in 16a.

Script

"../bin/opinimin" 39a≡

```
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
rootDir=$modulesdir/VUA-opinion-miner
cd $rootDir
export PATH=$PATH:.
python classify_kaf_naf_file.py -m $rootDir/final_models/nl/news_cfg1
```

◇

Uses: PATH 5b.

5 Utilities

5.1 Test script

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

"../bin/test" 39b≡

```
#!/bin/bash
ROOT=/home/huygen/projecten/pipelines/nlpp
TESTDIR=$ROOT/test
BIND=$ROOT/bin
mkdir -p $TESTDIR
cd $TESTDIR
[ $spotlightrunning ] || source /home/huygen/projecten/pipelines/nlpp/bin/start-
spotlight
cat $ROOT/nuweb/testin.naf    | $BIND/tok                    > $TEST-
DIR/test.tok.naf
cat test.tok.naf              | $BIND/mor                    > $TEST-
DIR/test.mor.naf
cat test.mor.naf              | $BIND/nerc_conll02 > $TESTDIR/test.nerc.naf
cat $TESTDIR/test.nerc.naf    | $BIND/wsd                    > $TEST-
DIR/test.wsd.naf
cat $TESTDIR/test.wsd.naf     | $BIND/ned                    > $TEST-
DIR/test.ned.naf
cat $TESTDIR/test.ned.naf     | $BIND/heideltime             > $TEST-
DIR/test.times.naf
cat $TESTDIR/test.times.naf   | $BIND/onto                   > $TEST-
DIR/test.onto.naf
cat $TESTDIR/test.onto.naf    | $BIND/srl                    > $TEST-
DIR/test.srl.naf
cat $TESTDIR/test.srl.naf     | $BIND/evcoref       > $TESTDIR/test.ecrf.naf
cat $TESTDIR/test.ecrf.naf    | $BIND/framesrl      > $TESTDIR/test.fsrl.naf
cat $TESTDIR/test.fsrl.naf    | $BIND/dbpner        > $TESTDIR/test.dbpner.naf
cat $TESTDIR/test.dbpner.naf  | $BIND/nomevent      > $TESTDIR/test.nomev.naf
cat $TESTDIR/test.nomev.naf   | $BIND/postsrl                > $TEST-
DIR/test.psrl.naf
cat $TESTDIR/test.psrl.naf    | $BIND/opinimin         > $TESTDIR/test.opin.naf
```

◇

Uses: nuweb 49b.

5.2 Logging

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

⟨ *variables of install-modules* 40a ⟩ ≡
```
LOGLEVEL=1
```
◇
Fragment referenced in 15b.

⟨ *logmess* 40b ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
```
◇
Fragment referenced in 6c, 7a, 14c, 17b, 40c.

5.3 Misc

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

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

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

⟨ *install from tarball* 40c ⟩ ≡
```
SUCCES=0
cd $modulesdir
⟨ move module (40d $DIR ) 6a ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (40e Installed $DIR ) 40b ⟩
  ⟨ remove old module (40f $DIR ) 6b ⟩
else
  ⟨ re-instate old module (40g $DIR ) 6c ⟩
fi
```
◇
Fragment never referenced.

A How to read and translate this document

This document is an example of *literate programming* [1]. It contains the code of all sorts of scripts and programs, combined with explaining texts. In this document the literate programming

tool `nuweb` is used, that is currently available from Sourceforge (URL:`nuweb.sourceforge.net`). The advantages of Nuweb are, that it can be used for every programming language and scripting language, that it can contain multiple program sources and that it is very simple.

A.1 Read this document

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

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

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

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

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

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

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

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

A.2 Process the document

The raw document is named `a_nlpp.w`. Figure 2 shows pathways to translate it into printable/viewable documents and to extract the program sources. Table 3 lists the tools that are

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

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

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

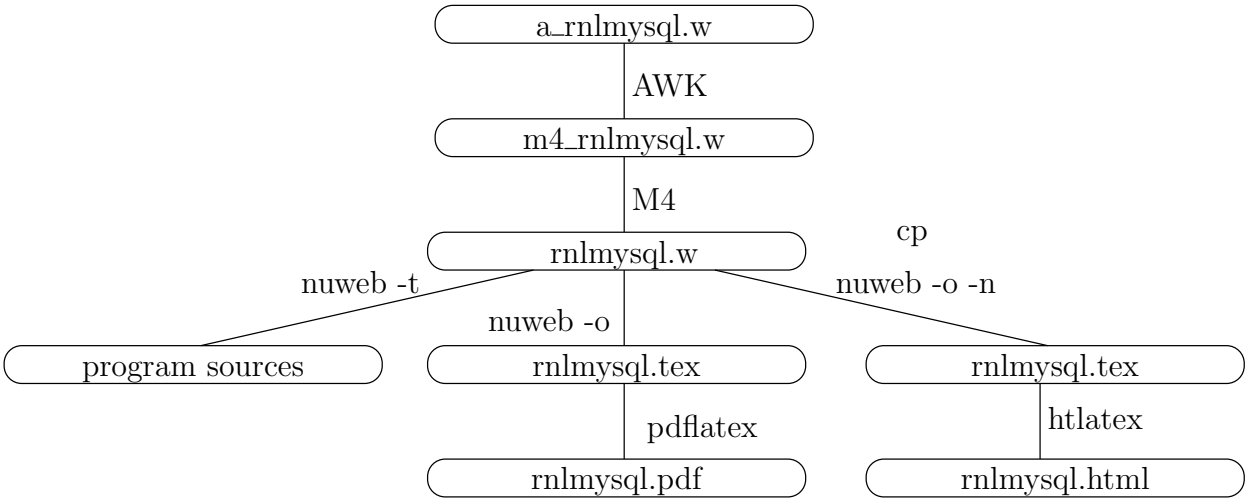

Figure 2: *Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.*

⟨ *parameters in Makefile* 42a ⟩ ≡
```
NUWEB=../env/bin/nuweb
```
◇
Fragment defined by 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Uses: `nuweb` 49b.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

`"Makefile"` 42b≡
⟨ *default target* 42c ⟩

⟨ *parameters in Makefile* 42a, ... ⟩

⟨ *impliciete make regels* 46a, ... ⟩
⟨ *expliciete make regels* 43d, ... ⟩
⟨ *make targets* 43a, ... ⟩
◇

The default target of make is `all`.

⟨ *default target* 42c ⟩ ≡
```
all : ⟨ all targets 43b ⟩
.PHONY : all
```
◇
Fragment referenced in 42b.
Defines: `all` 23b, `PHONY` 46b.

⟨ *make targets* 43a ⟩ ≡
```
clean:
        ⟨ clean up 9d, ... ⟩

◇
```
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.

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

⟨ *all targets* 43b ⟩ ≡
```
nlpp.pdf◇
```
Fragment referenced in 42c.
Uses: pdf 47a.

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* 43c ⟩ ≡
```
.SUFFIXES: .pdf .w .tex .html .aux .log .php

◇
```
Fragment defined by 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Defines: SUFFIXES Never used.
Uses: pdf 47a.

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* 43d ⟩ ≡
```
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 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.
Uses: nuweb 49b.

⟨ *clean up* 44a ⟩ ≡
```
rm -rf ../nuweb-1.58
```
◇
Fragment defined by 9d, 10c, 17f, 35g, 44a.
Fragment referenced in 43a.
Uses: nuweb 49b.

⟨ *expliciete make regels* 44b ⟩ ≡
```
../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 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.
Uses: nuweb 49b.

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* 44c ⟩ ≡
```
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 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.
Uses: print 47a.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 44d ⟩ ≡
```
nlpp.w : m4_nlpp.w inst.m4
        m4 -P m4_nlpp.w > nlpp.w
```
◇
Fragment defined by 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.

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* 45a ⟩ ≡

```
FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Defines: `FIGFILES` 45b.

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* 45b ⟩ ≡

```
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 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 47b, `PDF_FIG_NAMES` 47b, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 45a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 46a ⟩ ≡

```
%.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 46a, 50c.
Fragment referenced in 42b.
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* 46b ⟩ ≡

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

.PHONY : bibfile
```

◇

Fragment defined by 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.
Uses: `PHONY` 42c.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 47a ⟩ ≡
```
pdf : nlpp.pdf

print : nlpp.pdf
        lpr nlpp.pdf

view : nlpp.pdf
        evince nlpp.pdf

◇
```
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.
Defines: `pdf` 43bc, 47b, `print` 11a, 12ef, 14b, 32a, 44c, `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* 47b ⟩ ≡
```
nlpp.pdf : nlpp.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

◇
```
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.
Uses: `pdf` 47a, `PDFT_NAMES` 45b, `PDF_FIG_NAMES` 45b.

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* 47c ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 4abcd, 9afg, 13e, 47c.
Fragment referenced in 53a.
Uses: `nuweb` 49b.

⟨ *parameters in Makefile* 47d ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Uses: `nuweb` 49b.

⟨ *expliciete make regels* 48a ⟩ ≡
```
$(W2PDF) : nlpp.w $(NUWEB)
        $(NUWEB) nlpp.w
```
◇
Fragment defined by 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.

"../nuweb/bin/w2pdf" 48b≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20151127 at 1002h: Generated by nuweb from a_nlpp.w
NUWEB=../env/bin/nuweb
LATEXCOMPILER=pdflatex
```
⟨ *filenames in nuweb compile script* 48d ⟩
⟨ *compile nuweb* 48c ⟩

◇
Uses: nuweb 49b.

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* 48c ⟩ ≡
```
NUWEB=/home/huygen/projecten/pipelines/nlpp/env/bin/nuweb
```
⟨ *run the processors until the aux file remains unchanged* 49c ⟩
⟨ *remove the copy of the aux file* 49a ⟩
◇
Fragment referenced in 48b.
Uses: nuweb 49b.

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* 48d ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
```
◇
Fragment referenced in 48b.
Defines: auxfil 49c, 51c, 52a, indexfil 49c, 51c, nufil 49b, 51c, 52b, oldaux 49ac, 51c, 52a, oldindexfil 49c, 51c, texfil 49b, 51c, 52b, trunk 49b, 51c, 52bc.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 49a ⟩ ≡

```
rm $oldaux
◇
```

Fragment referenced in 48c, 51b.
Uses: `oldaux` 48d, 51c.

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* 49b ⟩ ≡

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

Fragment referenced in 49c.
Defines: `bibtex` 52bc, `makeindex` 52bc, `nuweb` 5a, 39b, 42a, 43d, 44ab, 47cd, 48bc, 50a, 51a.
Uses: `nufil` 48d, 51c, `texfil` 48d, 51c, `trunk` 48d, 51c.

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* 49c ⟩ ≡

```
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 49b ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
◇
```

Fragment referenced in 48c.
Uses: `auxfil` 48d, 51c, `indexfil` 48d, `oldaux` 48d, 51c, `oldindexfil` 48d.

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* 50a ⟩ ≡
```
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 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Uses: `nuweb` 49b.

Make the directory:

⟨ *expliciete make regels* 50b ⟩ ≡
```
$(htmldir) :
        mkdir -p $(htmldir)
```
◇

Fragment defined by 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.

The rule to copy files in it:

⟨ *impliciete make regels* 50c ⟩ ≡
```
$(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
```
◇

Fragment defined by 46a, 50c.
Fragment referenced in 42b.

Do the work:

⟨ *expliciete make regels* 50d ⟩ ≡
```
$(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
```
◇

Fragment defined by 43d, 44bcd, 46b, 48a, 50bd.
Fragment referenced in 42b.

Invoke:

⟨ *make targets* 50e ⟩ ≡
```
htm : $(htmldir) $(htmltarget)
```
◇

Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.

Create a script that performs the translation.

```
"w2html" 51a≡
        #!/bin/bash
        # w2html -- make a html file from a nuweb file
        # usage: w2html [filename]
        #  [filename]: Name of the nuweb source file.
        # 20151127 at 1002h: Generated by nuweb from a_nlpp.w
        echo "translate " $1 >w2html.log
        NUWEB=/home/huygen/projecten/pipelines/nlpp/env/bin/nuweb
        ⟨ filenames in w2html 51c ⟩

        ⟨ perform the task of w2html 51b ⟩

        ◇
```

Uses: nuweb 49b.

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 51b ⟩ ≡
        ⟨ run the html processors until the aux file remains unchanged 52a ⟩
        ⟨ remove the copy of the aux file 49a ⟩
        ◇
```

Fragment referenced in 51a.

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 51c ⟩ ≡
        nufil=$1
        trunk=${1%%.*}
        texfil=${trunk}.tex
        auxfil=${trunk}.aux
        oldaux=old.${trunk}.aux
        indexfil=${trunk}.idx
        oldindexfil=old.${trunk}.idx
        ◇
```

Fragment referenced in 51a.
Defines: `auxfil` 48d, 49c, 52a, `nufil` 48d, 49b, 52b, `oldaux` 48d, 49ac, 52a, `texfil` 48d, 49b, 52b, `trunk` 48d, 49b, 52bc.
Uses: `indexfil` 48d, `oldindexfil` 48d.

⟨ *run the html processors until the aux file remains unchanged* 52a ⟩ ≡

```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 52b ⟩
done
⟨ run tex4ht 52c ⟩

◇
```

Fragment referenced in 51b.
Uses: `auxfil` 48d, 51c, `oldaux` 48d, 51c.

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

⟨ *run the html processors* 52b ⟩ ≡

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

Fragment referenced in 52a.
Uses: `bibtex` 49b, `makeindex` 49b, `nufil` 48d, 51c, `texfil` 48d, 51c, `trunk` 48d, 51c.

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* 52c ⟩ ≡

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

Fragment referenced in 52a.
Uses: `bibtex` 49b, `makeindex` 49b, `trunk` 48d, 51c.

A.7 Create the program sources

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

⟨ *parameters in Makefile* 52d ⟩ ≡

```
MKDIR = mkdir -p

◇
```

Fragment defined by 42a, 43c, 45ab, 47d, 50a, 52d.
Fragment referenced in 42b.
Defines: `MKDIR` 53a.

⟨ *make targets* 53a ⟩ ≡
```
DIRS = ⟨ directories to create 4a, ... ⟩

$(DIRS) :
        $(MKDIR) $@
```
◇
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.
Defines: DIRS 53c.
Uses: MKDIR 52d.

⟨ *make scripts executable* 53b ⟩ ≡
```
chmod -R 775  ../bin/*
chmod -R 775  ../env/bin/*
```
◇
Fragment defined by 16b, 53b.
Fragment referenced in 53c.

⟨ *make targets* 53c ⟩ ≡
```
sources : nlpp.w $(DIRS) $(NUWEB)
        $(NUWEB) nlpp.w
        ⟨ make scripts executable 16b, ... ⟩
```
◇
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.
Uses: DIRS 53a.

A.8 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* 53d ⟩ ≡
```
transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant
```
◇
Fragment defined by 43a, 47ab, 50e, 53acd.
Fragment referenced in 42b.

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

```
"../env/bin/transplant" 54≡
    #!/bin/bash
    LOGLEVEL=1
    ⟨ set variables that point to the directory-structure 5a, ... ⟩
    ⟨ set paths after transplantation 12a, ... ⟩
    ⟨ re-install modules after the transplantation 20c ⟩

    ◇
```

B References

B.1 Literature

References

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

C Indexes

C.1 Filenames

C.2 Macro's

⟨ check this first 7f, 16c ⟩ Referenced in 15b.
⟨ check whether mercurial is present 17a ⟩ Referenced in 16c.
⟨ check/install the correct version of python 11a ⟩ Referenced in 10d.
⟨ clean up 9d, 10c, 17f, 35g, 44a ⟩ Referenced in 43a.
⟨ clone the heideltime wrapper 32c ⟩ Referenced in 32b.
⟨ compile nuweb 48c ⟩ Referenced in 48b.
⟨ compile the heideltime wrapper 33c ⟩ Referenced in 32b.
⟨ compile the nerc jar 26c ⟩ Referenced in 26b.
⟨ create a virtual environment for Python 13b ⟩ Referenced in 10d.
⟨ create javapython script 8b ⟩ Referenced in 15b.
⟨ default target 42c ⟩ Referenced in 42b.
⟨ directories to create 4abcd, 9afg, 13e, 47c ⟩ Referenced in 53a.
⟨ download svm models 28c ⟩ Referenced in 28a.
⟨ expliciete make regels 43d, 44bcd, 46b, 48a, 50bd ⟩ Referenced in 42b.
⟨ filenames in nuweb compile script 48d ⟩ Referenced in 48b.
⟨ filenames in w2html 51c ⟩ Referenced in 51a.
⟨ get jvntextpro-2.0.jar 33d ⟩ Referenced in 33c.
⟨ get or have 8a ⟩ Referenced in 9b, 11b, 17b, 20a, 21a, 27a, 28c, 29a, 30c, 33a, 35e, 36b, 37c, 38a.
⟨ get the nerc models 27a ⟩ Referenced in 26b.
⟨ impliciete make regels 46a, 50c ⟩ Referenced in 42b.
⟨ install ActivePython 11b ⟩ Referenced in 11a.
⟨ install Alpino 17b ⟩ Referenced in 15b.
⟨ install coreference-base 25c ⟩ Referenced in 16a.
⟨ install CRFsuite 24a ⟩ Referenced in 15b.
⟨ install from github 7a ⟩ Referenced in 25ac, 28a, 29d, 32c, 34c, 37a.
⟨ install from tarball 40c ⟩ Not referenced.
⟨ install kafnafparserpy 14c ⟩ Referenced in 10d.
⟨ install liblblbfgs 23c ⟩ Referenced in 24a.
⟨ install maven 10ab ⟩ Referenced in 15b.
⟨ install python packages 15a ⟩ Referenced in 10d.
⟨ install svm lib 28b ⟩ Referenced in 28a.
⟨ install SVMLight 23b ⟩ Referenced in 15b.
⟨ install the dbpedia-ner module 37a ⟩ Referenced in 16a.
⟨ install the event-coreference module 36b ⟩ Referenced in 16a.
⟨ install the heideltime module 32b ⟩ Referenced in 16a.
⟨ install the lu2synset converter 29a ⟩ Referenced in 16a.
⟨ install the morphosyntactic parser 25a ⟩ Referenced in 16a.
⟨ install the NERC module 26b ⟩ Referenced in 16a.
⟨ install the nomevent module 37c ⟩ Referenced in 16a.
⟨ install the onto module 30c ⟩ Referenced in 16a.
⟨ install the opinion-miner 38ad ⟩ Referenced in 16a.
⟨ install the post-SRL module 35e ⟩ Referenced in 16a.
⟨ install the Spotlight server 21ac ⟩ Referenced in 15b.
⟨ install the srl module 34c ⟩ Referenced in 16a.
⟨ install the ticcutils utility 19c ⟩ Referenced in 15b, 20c.
⟨ install the timbl utility 19d ⟩ Referenced in 15b, 20c.
⟨ install the tokenizer 24b ⟩ Referenced in 16a.
⟨ install the treetagger utility 18abcde, 19ab ⟩ Referenced in 15b.
⟨ install the WSD module 28a ⟩ Referenced in 16a.
⟨ install the NED module 29d ⟩ Referenced in 16a.
⟨ install VUA-pylib 23a ⟩ Referenced in 15b.
⟨ logmess 40b ⟩ Referenced in 6c, 7a, 14c, 17b, 40c.
⟨ make scripts executable 16b, 53b ⟩ Referenced in 53c.
⟨ make targets 43a, 47ab, 50e, 53acd ⟩ Referenced in 42b.
⟨ move module 6a ⟩ Referenced in 7a, 14c, 40c.
⟨ parameters in Makefile 42a, 43c, 45ab, 47d, 50a, 52d ⟩ Referenced in 42b.
⟨ perform the task of w2html 51b ⟩ Referenced in 51a.
⟨ put Antske's material in the heideltime wrapper 33a ⟩ Referenced in 32b.

⟨ put spotlight jar in the Maven repository 30a ⟩ Referenced in 29d.
⟨ re-install modules after the transplantation 20c ⟩ Referenced in 54.
⟨ re-instate old module 6c ⟩ Referenced in 7a, 14c, 40c.
⟨ remove old module 6b ⟩ Referenced in 7a, 14c, 40c.
⟨ remove outdated heideltime jars 34a ⟩ Referenced in 33e.
⟨ remove the copy of the aux file 49a ⟩ Referenced in 48c, 51b.
⟨ run tex4ht 52c ⟩ Referenced in 52a.
⟨ run the html processors 52b ⟩ Referenced in 52a.
⟨ run the html processors until the aux file remains unchanged 52a ⟩ Referenced in 51b.
⟨ run the processors until the aux file remains unchanged 49c ⟩ Referenced in 48c.
⟨ run the three processors 49b ⟩ Referenced in 49c.
⟨ set alpinohome 17e ⟩ Referenced in 25b.
⟨ set paths after transplantation 12a, 13a, 14b ⟩ Referenced in 54.
⟨ set up java 9be ⟩ Referenced in 15b.
⟨ set up python 10d ⟩ Referenced in 15b.
⟨ set variables that point to the directory-structure 5ab ⟩ Referenced in 5c, 15b, 54.
⟨ start the Spotlight server on localhost 22f ⟩ Referenced in 22a.
⟨ test whether spotlighthost runs 22e ⟩ Referenced in 22a.
⟨ test whether virtualenv is present on the host 13c ⟩ Referenced in 13b.
⟨ try to obtain a running spotlightserver 22a ⟩ Referenced in 21d.
⟨ unpack ticcutils or timbl 20a ⟩ Referenced in 19cd.
⟨ variables of install-modules 40a ⟩ Referenced in 15b.
⟨ write script chasbang.awk 12e ⟩ Referenced in 12a.
⟨ write script tran 12d ⟩ Referenced in 12a.

C.3 Variables