

Standardised Dutch NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

15th December 2015
09:08 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	2
1.1	List of the modules to be installed	3
1.2	The things that are not open-source yet	4
1.3	Multi-linguality	4
1.4	File-structure of the pipeline	4
2	How to obtain modules and other material	6
2.1	Location-dependency	6
2.2	Reversible update	6
2.3	Installation from Github	7
2.4	Installation from the snapshot	7
3	Java and Python environment	8
3.1	Java	8
3.2	Maven	9
3.3	Python	10
3.3.1	Virtual environment	12
3.3.2	Transplant the virtual environment	13
3.3.3	KafNafParserPy	13
3.3.4	Python packages	13
4	Installation of the modules	14
4.1	The installation script	14
4.2	Check availability of resources	16
4.3	Install utilities and resources	17
4.3.1	Language detection	17
4.3.2	Alpino	18
4.3.3	Treetagger	19
4.3.4	Timbl and Ticcutils	20
4.3.5	Spotlight	21
4.3.6	VUA-pylib	24
4.3.7	SVMLight	24
4.3.8	CRFsuite	25
4.4	Install modules	25

4.4.1	Install tokenizer	25
4.4.2	Morphosyntactic parser	26
4.4.3	Nominal coreference-base	27
4.4.4	Named entity recognition (NERC)	28
4.4.5	Wordsense-disambiguation	29
4.4.6	Lexical-unit converter	30
4.4.7	NED	31
4.4.8	Ontotagger	32
4.4.9	Framenet SRL	33
4.4.10	Heideltime	34
4.4.11	Semantic Role labelling	36
4.4.12	SRL postprocessing	37
4.4.13	Event coreference	38
4.4.14	Dbpedia-ner	38
4.4.15	Nominal events	39
4.4.16	Opinion miner	40
5	Utilities	41
5.1	Test script	41
5.2	Logging	42
5.3	Misc	43
A	How to read and translate this document	43
A.1	Read this document	44
A.2	Process the document	44
A.3	The Makefile for this project.	45
A.4	Get Nuweb	46
A.5	Pre-processing	47
A.5.1	Process ‘dollar’ characters	47
A.5.2	Run the M4 pre-processor	47
A.6	Typeset this document	47
A.6.1	Figures	48
A.6.2	Bibliography	49
A.6.3	Create a printable/viewable document	49
A.6.4	Create HTML files	52
A.7	Create the program sources	55
A.8	Restore paths after transplantation	56
B	References	57
B.1	Literature	57
C	Indexes	57
C.1	Filenames	57
C.2	Macro’s	57
C.3	Variables	59

1 Introduction

This document describes the current set-up of 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.

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 successfully 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	807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184	mor
NERC	4.4.4	Gith./snap	5cacac28fcaa6e91d5f2a4cc9b486b24ac163641	nerc
WSD	4.4.5	Gith./snap	030043903b42f77cd20a9b2443de137e2efe8513	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		opimin
Framenet SRL	4.4.9	snapshot		fsrl
Dbpedia_ner	4.4.14	Github	ab1dcdb860f0ff29bc979f646dc382122a101fc2	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	3.3.3	Github
Alpino	20706	4.3.2	RUG
Ticcutils	0.7	4.3.4	ILK
Timbl	6.4.6	4.3.4	ILK
Treetagger	3.2	4.3.3	Uni. München
Spotlight server	0.7	4.3.5	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.

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

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. However, 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

This version of the pipeline is multi-lingual, i.e. it can annotate Dutch 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

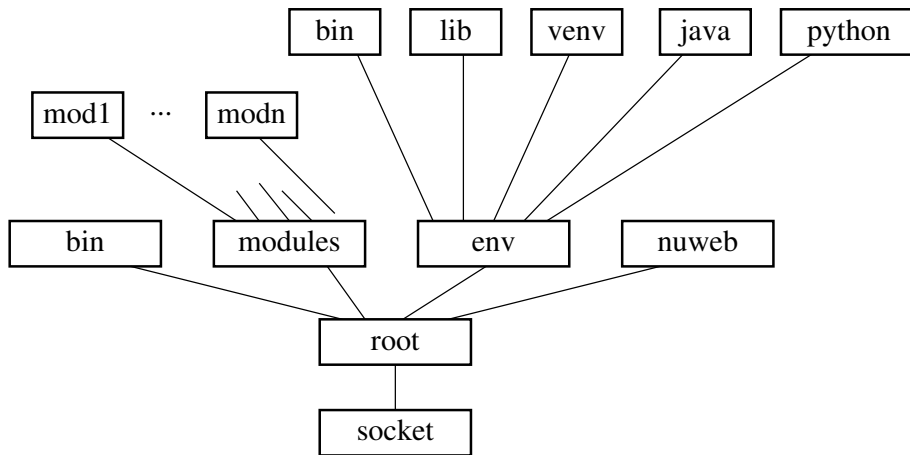


Figure 1: Directory-structure of the pipeline (see text).

directories have the following 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.

$\langle \text{directories to create} \rangle \equiv$
`../modules` \diamond

Fragment defined by 4, 5abc, 9aef, 12d, 50c.

Fragment referenced in 56a.

```

< directories to create 5a > ≡
    ../bin ../env/bin ◇

```

Fragment defined by 4, 5abc, 9aef, 12d, 50c.
 Fragment referenced in 56a.

```

< directories to create 5b > ≡
    ../env/lib ◇

```

Fragment defined by 4, 5abc, 9aef, 12d, 50c.
 Fragment referenced in 56a.

```

< directories to create 5c > ≡
    ../env/etc ◇

```

Fragment defined by 4, 5abc, 9aef, 12d, 50c.
 Fragment referenced in 56a.

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 5d > ≡
    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 5de, 7h.
 Fragment referenced in 6a, 15, 57.
 Uses: nuweb 52b.

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

```

< set variables that point to the directory-structure 5e > ≡
    export PATH=$envbindir:$PATH
    ◇

```

Fragment defined by 5de, 7h.
 Fragment referenced in 6a, 15, 57.
 Defines: PATH 9d, 10c, 41.

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

```

"../env/bin/progenv" 6a≡
  #!/bin/bash
  < set variables that point to the directory-structure 5d, ... >
  export progenvset=0
  ◇

```

File defined by 6a, 8d.

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” actually 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 6b > ≡
  if
    [ -e @1 ]
  then
    mv @1 old.@1
  fi
  ◇

```

Fragment referenced in 7c, 13b, 43b.

```

< remove old module 6c > ≡
  rm -rf old.@1
  ◇

```

Fragment referenced in 7c, 13b, 43b.

```

< re-instate old module 7a > ≡
    mv old.@1 @1
    MESS="Replaced previous version of @1"
    < logmess (7b $MESS ) 43a >

```

◇

Fragment referenced in 7c, 13b, 43b.

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 7c > ≡
    cd $modulesdir
    < move module (7d $DIRN ) 6b >
    git clone $GITU
    if
        [ $? -gt 0 ]
    then
        < logmess (7e Cannot install current $MODNAM version ) 43a >
        < re-instate old module (7f $DIRN ) 7a >
    else
        < remove old module (7g $DIRN ) 6c >
        cd $modulesdir/$DIRN
        git checkout $GITC
    fi

```

◇

Fragment referenced in 26c, 27b, 29c, 31d, 34c, 36c, 39a.

2.4 Installation from the snapshot

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

```

< set variables that point to the directory-structure 7h > ≡
    if
        [ ! $snapshotsocket ]
    then
        export snapshotsocket=/home/huygen/projecten/pipelines
    fi

```

◇

Fragment defined by 5de, 7h.

Fragment referenced in 6a, 15, 57.

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 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 8a > ≡
  if
    [ ! -e $pipesocket/nrkey ]
  then
    echo "No key to connect to snapshot!"
    exit 1
  fi
◇
```

Fragment defined by [8a](#), [16c](#).

Fragment referenced in [15](#).

Update the local snapshot repository.

```
< get the snapshot 8b > ≡
  cd $snapshotsocket
  rsync -e "ssh -i $pipesocket/nrkey" -r newsreader@kyoto:t_nlpp_resources .
◇
```

Fragment referenced in [15](#).

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 8c > ≡
  echo '#!/bin/bash' > /home/huygen/projecten/pipelines/nlpp/env/bin/javapython
◇
```

Fragment referenced in [15](#).

Cause the module scripts to read the javapython script.

```
"../env/bin/progenv" 8d ≡
  source $envbindir/javapython
◇
```

File defined by [6a](#), [8d](#).

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 > \equiv
`../env/java` \diamond

Fragment defined by [4](#), [5abc](#), [9aef](#), [12d](#), [50c](#).
 Fragment referenced in [56a](#).

< set up java 9b > \equiv

```
if
  [ ! $java_installed ]
then
  cd $envdir/java
  tar -xzf $snapshotsocket/t_nlpp_resources/server-jre-7u72-linux-x64.tar.gz
  echo "ex-
port java_installed=0" >> /home/huygen/projecten/pipelines/nlpp/installed_modules
fi

```

 \diamond

Fragment defined by [9bd](#).
 Fragment referenced in [15](#).

Remove the java-ball when cleaning up:

< clean up 9c > \equiv

```
rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz

```

 \diamond

Fragment defined by [9c](#), [10d](#), [19a](#), [37g](#), [46d](#).
 Fragment referenced in [45c](#).

Set variables for Java.

< set up java 9d > \equiv

```
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

```

 \diamond

Fragment defined by [9bd](#).
 Fragment referenced in [15](#).
 Uses: [PATH 5e](#).

Put jars in the jar subdirectory of the java directory:

< directories to create 9e > \equiv
`../env/java/jars` \diamond

Fragment defined by [4](#), [5abc](#), [9aef](#), [12d](#), [50c](#).
 Fragment referenced in [56a](#).

3.2 Maven

Some Java-based modules can best be compiled with [Maven](#).

< directories to create 9f > \equiv
`../env/apache-maven-3.0.5` \diamond

Fragment defined by [4](#), [5abc](#), [9aef](#), [12d](#), [50c](#).
 Fragment referenced in [56a](#).

```

< install maven if it hasn't been done 10a > ≡
    if
        [ ! $maven_installed ]
    then
        < install maven 10b, ... >
        export maven_installed=0
        echo "ex-
port maven_installed=0" >> /home/huygen/projecten/pipelines/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 15.

```

< install maven 10b > ≡
    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 10bc.

Fragment referenced in 10a.

```

< install maven 10c > ≡
    export MAVEN_HOME=$envdir/apache-maven-3.0.5
    export PATH=${MAVEN_HOME}/bin:${PATH}
    ◇

```

Fragment defined by 10bc.

Fragment referenced in 10a.

Uses: PATH 5e.

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

```

< clean up 10d > ≡
    rm -rf ../env/apache-maven-3.0.5
    ◇

```

Fragment defined by 9c, 10d, 19a, 37g, 46d.

Fragment referenced in 45c.

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 will use Python 2.7 if it 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.

```

< install python if it hasn't been done 11a > ≡
if
  [ ! $python_installed ]
then
  < set up python 11b >
  export python_installed=0
  echo "ex-
port python_installed=0" >> /home/huygen/projecten/pipelines/nlpp/installed_modules
else
  < activate the python environment 12c, ... >
fi
◇

```

Fragment never referenced.

```

< set up python 11b > ≡
  < check/install the correct version of python 11c >
  < create a virtual environment for Python 12a >
  < activate the python environment 12c, ... >
  < install kafnaparserpy 13b >
  < install python packages 14a >
◇

```

Fragment referenced in 11a, 15.

```

< check/install the correct version of python 11c > ≡
pythonok='python --
version 2>&1 | gawk '{if(match($2, "2.7")) print "yes" ; else print "no" }'
if
  [ "$pythonok" == "no" ]
then
  < install ActivePython 11d >
fi
◇

```

Fragment referenced in 11b.

Defines: pythonok Never used.

Uses: print 50a.

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 11d > ≡
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 pip virtualenv setuptools
◇

```

Fragment referenced in 11c.

Uses: virtualenv 12b.

3.3.1 Virtual environment

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

```
< create a virtual environment for Python 12a > ≡
  < test whether virtualenv is present on the host 12b >
    cd $envdir
    virtualenv venv
  ◇
```

Fragment referenced in 11b.

Uses: `virtualenv` 12b.

```
< test whether virtualenv is present on the host 12b > ≡
  which virtualenv
  if
    [ $? -ne 0 ]
  then
    echo Please install virtualenv
    exit 1
  fi
  ◇
```

Fragment referenced in 12a.

Defines: `virtualenv` 11d, 12a.

```
< activate the python environment 12c > ≡
  source $envdir/venv/bin/activate
  echo 'source $en-
vdir/venv/bin/activate' >> /home/huygen/projecten/pipelines/nlpp/env/bin/javapython
  ◇
```

Fragment defined by 12ce.

Fragment referenced in 11ab.

Defines: `activate` 13a.

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

```
< directories to create 12d > ≡
  ../env/python ◇
```

Fragment defined by 4, 5abc, 9aef, 12d, 50c.

Fragment referenced in 56a.

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

```
< activate the python environment 12e > ≡
  echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /home/huygen/projecten/pipelines/nlpp/env/bin/javapyt
export PYTHONPATH=$envdir/python:$PYTHONPATH
  ◇
```

Fragment defined by 12ce.

Fragment referenced in 11ab.

Defines: `PYTHONPATH` Never used.

3.3.2 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 13a > ≡
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 57.

Uses: activate 12c, print 50a.

3.3.3 KafNafParserPy

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

```
< install kafnafparserpy 13b > ≡
cd $envdir/python
DIRN=KafNafParserPy
< move module (13c $DIRN ) 6b >
git clone https://github.com/cltl/KafNafParserPy.git
if
[ $? -gt 0 ]
then
< logmess (13d Cannot install current $DIRN version ) 43a >
< re-instate old module (13e $DIRN ) 7a >
else
< remove old module (13f $DIRN ) 6c >
fi
◇
```

Fragment referenced in 11b.

3.3.4 Python packages

Install python packages:

lxml:

pyyaml: for coreference-graph

```

< install python packages 14a > ≡
    pip install lxml
    pip install pyyaml
    ◇

```

Fragment referenced in 11b.

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`. It maintains a list of the modules and utilities that it 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 that sets Bash variables. It ought to be sourced.

```

< read the list of installed modules 14b > ≡
    if
        [ -e /home/huygen/projecten/pipelines/nlpp/installed_modules ]
    then
        source /home/huygen/projecten/pipelines/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 15.

The first part of the script installs the utilities:

```

"../bin/install-modules" 15≡
  #!/bin/bash
  echo Set up environment
  < set variables that point to the directory-structure 5d, ... >
  < get the snapshot 8b >
  < read the list of installed modules 14b >
  < variables of install-modules 42c >
  < check this first 8a, ... >
  < create javapython script 8c >
  echo ... Java
  < set up java 9b, ... >
  < install maven if it hasn't been done 10a >
  echo ... Python
  < set up python 11b >
  echo ... Alpino
  < install Alpino 18c >
  echo ... Spotlight
  < install the Spotlight server if it hasn't been done 22a >
  echo ... Treetagger
  < install the treetagger if it hasn't been done 19b >
  echo ... Ticcutils and Timbl
  < install the ticcutils utility 21a >
  < install the timbl utility 21b >
  echo ... VUA-pylib, SVMlight, CRFsuite
  < install VUA-pylib 24c >
  < install SVMLight 25a >
  < install CRFsuite 25b >

```

◇

File defined by 15, 16a.

Next, install the modules:

```

"../bin/install-modules" 16a≡
    echo Install modules
    echo ... Tokenizer
    <install the tokenizer 26a>
    echo ... Morphosyntactic parser
    <install the morphosyntactic parser 26c>
    echo ... NERC
    <install the NERC module 28a>
    echo ... Coreference base
    <install coreference-base 27b>
    echo ... WSD
    <install the WSD module 29c>
    echo ... Ontotagger
    <install the onto module 32c>
    echo ... Heideltime
    <install the heideltime module 34b>
    echo ... SRL
    <install the srl module 36c>
    echo ... NED
    <install the NED module 31d>
    echo ... Event-coreference
    <install the event-coreference module 38b>
    echo ... lu2synset
    <install the lu2synset converter 31a>
    echo ... dbpedia-ner
    <install the dbpedia-ner module 39a>
    echo ... nominal event
    <install the nomevent module 39c>
    <install the post-SRL module 37e>
    <install the opinion-miner 40a, ... >

    echo Final
    ◇

```

File defined by 15, 16a.

```

<make scripts executable 16b>≡
    chmod 775 ../bin/install-modules
    ◇

```

Fragment defined by 16b, 17d, 56b.

Fragment referenced in 56c.

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 8a, 16c.

Fragment referenced in 15.


```

⟨ 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 27b.

4.3 Install utilities and resources

4.3.1 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 `lang`. All pipeline modules expect that this variable has been set.

```

"../env/bin/langdetect.py" 17b≡
    #!/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: lang 18a, print 50a.

```

"../bin/langdetect" 17c≡
    #!/bin/bash
    source $aenvbindir/progenv
    echo 'cat | python /home/huygen/projecten/pipelines/nlpp/env/bin/langdetect.py'
◇

```

```

⟨ make scripts executable 17d ⟩ ≡
    chmod 775 /home/huygen/projecten/pipelines/nlpp/bin/langdetect
◇

```

Fragment defined by 16b, 17d, 56b.

Fragment referenced in 56c.

```

⟨ set the language variable 18a ⟩ ≡
    naflang='echo cat @1 | /home/huygen/projecten/pipelines/nlpp/bin/langdetect'
    ◇

```

Fragment referenced in 42a.

Defines: lang 17b.

Currently, the pipeline understands only English and Dutch. The following macro aborts pipeline processing when the language is not English or Dutch.

```

⟨ abort when the language is not English or Dutch 18b ⟩ ≡
    if
        [ ! "$naflang" == 'nl' ] && [ ! "$naflang" == "en" ]
    then
        echo Language of NAF document not set.
        echo Set variable "naflang" to "en" of "nl" and try again.
        echo Aborting :-(
        exit 4
    fi
    ◇

```

Fragment referenced in 26b.

4.3.2 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 18c ⟩ ≡
    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/huygen/projecten/pipelines/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 15.

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 18d ⟩ ≡
    export ALPINO_HOME=$modulesdir/Alpino
    ◇

```

Fragment referenced in 26d.

Defines: ALPINO_HOME Never used.

Remove the tarball when cleaning up:

```

< clean up 19a > ≡
    rm -rf $snapshotsocket/snapshots/Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
    ◇

```

Fragment defined by 9c, 10d, 19a, 37g, 46d.

Fragment referenced in 45c.

4.3.3 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

```

< install the treetagger if it hasn't been done 19b > ≡
    if
        [ ! $treetagger_installed ]
    then
        < install the treetagger utility 19c, ... >
        export treetagger_installed=0
        echo "export treetag-
ger_installed=0" >> /home/huygen/projecten/pipelines/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 15.

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

```

< install the treetagger utility 19c > ≡
    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 19cde, 20abcd.

Fragment referenced in 19b.

The source tarball, scripts and the installation-script:

```

< install the treetagger utility 19d > ≡
    TREETAGSRC=tree-tagger-linux-3.2.tar.gz
    TREETAGSCRIPTS=tagger-scripts.tar.gz
    TREETAG_INSTALLSCRIPT=install-tagger.sh
    ◇

```

Fragment defined by 19cde, 20abcd.

Fragment referenced in 19b.

Parametersets:

```

< install the treetagger utility 19e > ≡
    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 19cde, 20abcd.

Fragment referenced in 19b.

Download everything in the target directory:

```

< install the treetagger utility 20a > ≡
    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 19cde, 20abcd.

Fragment referenced in 19b.

Run the install-script:

```

< install the treetagger utility 20b > ≡
    chmod 775 $TREETAG_INSTALLSCRIPT
    ./ $TREETAG_INSTALLSCRIPT
    ◇

```

Fragment defined by 19cde, 20abcd.

Fragment referenced in 19b.

Make the treetagger utilities available for everybody.

```

< install the treetagger utility 20c > ≡
    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 19cde, 20abcd.

Fragment referenced in 19b.

Remove the tarballs:

```

< install the treetagger utility 20d > ≡
    rm $TREETAGSRC
    rm $TREETAGSCRIPTS
    rm $TREETAG_INSTALLSCRIPT
    rm $DUTCHPARS_UTF_GZ
    rm $DUTCH_TAGSET
    rm $DUTCHPARS_2_GZ
    ◇

```

Fragment defined by 19cde, 20abcd.

Fragment referenced in 19b.

4.3.4 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 21a > ≡
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 21c >
◇

```

Fragment referenced in 15, 21d.

```

< install the timbl utility 21b > ≡
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
< unpack ticcutils or timbl 21c >
◇

```

Fragment referenced in 15, 21d.

```

< unpack ticcutils or timbl 21c > ≡
SUCCES=0
ticbeldir='mktemp -t -d tickbel.XXXXXX'
cd $ticbeldir
tar -xzf $snapshotsocket/t_nlpp_resources/$TARB
cd $DIR
./configure --prefix=$envdir
make
make install
cd $piperoot
rm -rf $ticbeldir
◇

```

Fragment referenced in 21ab.

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

```

< re-install modules after the transplantation 21d > ≡
< install the ticcutils utility 21a >
< install the timbl utility 21b >
◇

```

Fragment referenced in 57.

4.3.5 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 if it hasn't been done 22a > ≡
if
[ ! $spotlight_installed ]
then
  < install the Spotlight server 22b, ... >
  export spotlight_installed=0
  echo "export spot-
light_installed=0" >> /home/huygen/projecten/pipelines/nlpp/installed_modules
fi
```

◇

Fragment referenced in 15.

```
< install the Spotlight server 22b > ≡
cd $envdir
tar -xzf $snapshotsocket/t_nlpp_resources/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 22bc.

Fragment referenced in 22a.

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

```
< install the Spotlight server 22c > ≡
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 22bc.

Fragment referenced in 22a.

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 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" 23a≡
# 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 23b >
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 23b > ≡
< test whether spotlighthost runs (23c $spotlighthost ) 24a >
if
  [ ! $spotlightrunning ]
then
  if
    [ "$spotlighthost" != "localhost" ]
  then
    export spotlighthost=localhost
    < test whether spotlighthost runs (23d $spotlighthost ) 24a >
  fi
fi
if
  [ ! $spotlightrunning ]
then
  < start the Spotlight server on localhost 24b >
  < test whether spotlighthost runs (23e $spotlighthost ) 24a >
fi
if
  [ ! $spotlightrunning ]
then
  echo "Cannot start spotlight"
  exit 4
fi
◇

```

Fragment referenced in 23a.

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 24a > ≡
    exec 6<>/dev/tcp/@1/2060
    if
        [ $? -eq 0 ]
    then
        export spotlightrunning=0
    else
        spotlightrunning=
    fi
    exec 6<&-
    exec 6>&-
    ◇

```

Fragment referenced in 23b.

```

< start the Spotlight server on localhost 24b > ≡
    [ $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 23b.

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.

4.3.6 VUA-pylib

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

```

< install VUA-pylib 24c > ≡
    cd $envdir/python
    git clone https://github.com/cltl/VUA_pylib.git
    ◇

```

Fragment referenced in 15.

4.3.7 SVMlight

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

Installation goes like this:


```

< install SVMlight 25a > ≡
    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 15.

Uses: all 45b.

4.3.8 CRFsuite

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

```

< install CRFsuite 25b > ≡
    tempdir='mktemp -d -t crfsuite.XXXXXX'
    cd $tempdir
    tar -xzf $snapshotsocket/snapshots/crfsuite-0.12-x86_64.tar.gz
    cd crfsuite-0.12
    cp -r bin/crfsuite $envbindir/
    mkdir -p $envdir/include/
    cp -r include/* $envdir/include/
    mkdir -p $envdir/lib/
    cp -r lib/* $envdir/lib/
    cd m4_aprojdire
    rm -rf $tempdir
    ◇

```

Fragment referenced in 15.

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 26a > ≡
    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" 26b≡
    #!/bin/bash
    source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
    < abort when the language is not English or Dutch 18b >
    JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
    java -Xmx1000m -jar $JARFILE tok -l $naflang --inputkaf
    ◇

```

4.4.2 Morphosyntactic parser

Module

```

< install the morphosyntactic parser 26c > ≡
    MODNAM=morphosynparser
    DIRN=morphosyntactic_parser_nl
    GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
    GIRC=807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184
    < install from github 7c >
    cd $modulesdir/morphosyntactic_parser_nl
    git checkout 807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184
    ◇

```

Fragment referenced in 16a.

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" 26d≡
    #!/bin/bash
    source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
    < get the mor time-out parameter 27a >
    ROOT=$piperoot
    MODDIR=$modulesdir/morphosyntactic_parser_nl
    < set alpinohome 18d >
    cat | python $MODDIR/core/morph_syn_parser.py $timeoutarg
    ◇

```

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

```

⟨ get the mor time-out parameter 27a ⟩ ≡
    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
    fi
    ◇

```

Fragment referenced in 26d.

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 27b ⟩ ≡
    MODNAM=coreference-base
    DIRN=coreference-base
    GITU=https://github.com/opener-project/coreference-base.git
    GITC=bfa5aec0fa498e57fe14dd4d2c51365dd09a0757
    ⟨ install from github 7c ⟩
    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" 27c≡
    #!/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 28a > ≡
    < compile the nerc jar 28b >
    < get the nerc models 29a >
```

◇

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 28b > ≡
    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 28a.

The current version of the pipeline uses the following models, that have been made available 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 29a > ≡
  mkdir -p $modulesdir/nerc-models-nl
  cd $modulesdir/nerc-models-nl
  tar -xzf $snapshotsocket/snapshots/nerc-models-nl.tgz
  chmod -R 775 $modulesdir/nerc-models-nl
  ◇
```

Fragment referenced in 28a.

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

```
"../bin/nerc_conll02" 29b ≡
  #!/bin/bash
  source /home/huygen/projecten/pipelines/nlpp/env/bin/progen
  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 29c > ≡
  MODNAM=wsd
  DIRN=svm_wsd
  GITU=https://github.com/cltl/svm_wsd.git
  GITC=030043903b42f77cd20a9b2443de137e2efe8513
  < install from github 7c >
  cd $modulesdir/svm_wsd
  < install svm lib 30a >
  < download svm models 30b >
```

◇

Fragment referenced in 16a.

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

```
< install svm lib 30a > ≡
    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 29c.

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

```
< download svm models 30b > ≡
    cd $modulesdir
    #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 29c.

Script

```
"../bin/wsd" 30c≡
    #!/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 -ref odwnSY
    ◇
```

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 31a > ≡
  < get or have (31b lu2synset.tgz ) ? >
  cd $modulesdir
  tar -xzf $snapshotsocket/snapshots/lu2synset.tgz
  ◇

```

Fragment referenced in 16a.

Script

```

"../bin/lu2synset" 31c≡
  #!/bin/bash
  source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
  ROOT=$piperoot
  JAVAILIBDIR=$modulesdir/lexicalunitconvertor/lib
  RESOURCESDIR=$modulesdir/lexicalunitconvertor/resources
  JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
  java -Xmx812m -
  cp $JAVAILIBDIR/$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 31d > ≡
  < put spotlight jar in the Maven repository 32a >
  MODNAM=ned
  DIRN=ixa-pipe-ned
  GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
  GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
  < install from github 7c >
  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 32a> ≡
    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 31d.

Script

```

"../bin/ned" 32b≡
    #!/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 32c> ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/snapshots/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" 33≡
    #!/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 containing “frameMap.size()=...”. A GAWK script removes these lines.

```

"../bin/framesrl" 34a≡
    #!/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 50a](#).

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 34b> ≡
    moduledir=/home/huygen/projecten/pipelines/nlpp/modules/ixa-pipe-time
    <clone the heideltime wrapper 34c>
    <put Antske's material in the heideltime wrapper 35a>
    <compile the heideltime wrapper 35b>

```

Fragment referenced in [16a](#).

```

<clone the heideltime wrapper 34c> ≡
    MODNAM=heideltime
    DIRN=ixa-pipe-time
    GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
    GITC=da4604a7b33975e977017440cbc10f7d59917ddf
    <install from github (34d ixa-pipe-time) 7c>
    mkdir $moduledir/lib

```

Fragment referenced in [34b](#).

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 35a > ≡
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 34b.

Compile the Heideltime wrapper according to the [instruction](#) on Github.

```
< compile the heideltime wrapper 35b > ≡
< get jvntextpro-2.0.jar 35c >
< activate the install-to-project-repo utility 35d >
cd /home/huygen/projecten/pipelines/nlpp/modules/$DIRN
mvn clean install
◇
```

Fragment referenced in 34b.

```
< get jvntextpro-2.0.jar 35c > ≡
cd /home/huygen/projecten/pipelines/nlpp/modules/$DIRN/lib
wget http://ixa2.si.ehu.es/%7Ejibalar/jvntextpro-2.0.jar
◇
```

Fragment referenced in 35b.

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 35d > ≡
< remove outdated heideltime jars 36a >
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 35b.

```

⟨ remove outdated heideltime jars 36a ⟩ ≡
  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 35d.

Script

```

"../bin/heideltime" 36b≡
  #!/bin/bash
  source /home/huygen/projecten/pipelines/nlpp/env/bin/progen
  HEIDELDIR=$modulesdir/ixa-pipe-time
  cd $HEIDELDIR
  iconv -t utf-8//IGNORE | java -Xmx1000m -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 36c ⟩ ≡
  MODNAM=srl
  DIRN=vua-srl-nl
  GITU=https://github.com/newsreader/vua-srl-nl.git
  GITC=675d22d361289ede23df11dcdb17195f008c54bf
  ⟨ install from github 7c ⟩
  ◇

```

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" 36d≡
  #!/bin/bash
  source /home/huygen/projecten/pipelines/nlpp/env/bin/progen
  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 36d, 37abcd.

Create a feature-vector.

```

"../bin/srl" 37a≡
    cat | tee $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
    ◇

```

File defined by 36d, 37abcd.

Run the trained model on the feature-vector.

```

"../bin/srl" 37b≡
    timbl -m0:I1,2,3,4 -i 25Feb2015_e-mags_mags_press_newspapers.wgt -
    t $FEATUREVECTOR -o $TIMBLOUTPUTFILE >/dev/null 2>/dev/null
    ◇

```

File defined by 36d, 37abcd.

Insert the SRL values into the NAF file.

```

"../bin/srl" 37c≡
    python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
    ◇

```

File defined by 36d, 37abcd.

Clean up.

```

"../bin/srl" 37d≡
    rm -rf $TEMPDIR
    ◇

```

File defined by 36d, 37abcd.

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 37e ⟩ ≡
    ⟨ get or have (37f 20150706vua-srl-dutch-additional-roles.tgz ) ? ⟩
    cd $modulesdir
    tar -xzf $snapshotsocket/snapshots/20150706vua-srl-dutch-additional-roles.tgz
    ◇

```

Fragment referenced in 16a.

```

⟨ clean up 37g ⟩ ≡
    rm -rf $snapshotsocket/snapshots/20150706vua-srl-dutch-additional-roles.tgz
    ◇

```

Fragment defined by 9c, 10d, 19a, 37g, 46d.

Fragment referenced in 45c.

Script

```
"../bin/postsr1" 38a≡
#!/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 38b> ≡
cd $modulesdir
tar -xzf $snapshotsocket/snapshots/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" 38c≡
#!/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 than NER, because it checks DBpedia for the candidate NE-'s.

Module

```

< install the dbpedia-ner module 39a > ≡
MODNAM=dbpedia_ner
DIRN=dbpedia_ner
GITU=https://github.com/PaulHuygen/dbpedia_ner.git
GITC=ab1dcdbd860f0ff29bc979f646dc382122a101fc2
< install from github 7c >
◇

```

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" 39b ≡
#!/bin/bash
source /home/huygen/projecten/pipelines/nlpp/env/bin/progenv
[ $spotlightrunning ] || source /home/huygen/projecten/pipelines/nlpp/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 39c > ≡
cd $modulesdir
unzip -q $snapshotsocket/snapshots/vua-postprocess-nl.zip
◇

```

Fragment referenced in 16a.

Script

```

"../bin/nomevent" 39d ≡
#!/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 40a> ≡
  cd /home/huygen/projecten/pipelines/nlpp/modules
  tar -xzf /home/huygen/projecten/pipelines/20151012VUA-opinion-miner.tgz
  ◇
```

Fragment defined by 40ac.

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" 40b≡
  [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 40c> ≡
  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 40ac.

Fragment referenced in 16a.

Script


```

"../bin/opinimin" 41≡
    #!/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 5e.

5 Utilities

5.1 Test script

The following script pushes a test-document through the modules of the pipeline.

```

"../bin/test" 42a≡
    #!/bin/bash
    ROOT=/home/huygen/projecten/pipelines/nlpp
    TESTDIR=$ROOT/test
    TESTIN=$ROOT/nuweb/test.nl.in.naf
    if
        [ $1 == "en" ]
    then
        TESTIN=$ROOT/nuweb/test.en.in.naf
    fi
    BIND=$ROOT/bin
    mkdir -p $TESTDIR
    cd $TESTDIR
    [ $spotlightrunning ] || source /home/huygen/projecten/pipelines/nlpp/bin/start-
    spotlight
    < set the language variable (42b $TESTIN ) 18a >
    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/postsrsl        > $TEST-
    DIR/test.psrsl.naf
    cat $TESTDIR/test.psrsl.naf     | $BIND/opinimin        > $TESTDIR/test.opin.naf
    ◇

```

5.2 Logging

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

```

< variables of install-modules 42c > ≡
    LOGLEVEL=1
    ◇

```

Fragment referenced in 15.

```

< logmess 43a > ≡
    if
        [ $LOGLEVEL -gt 0 ]
    then
        echo @1
    fi
◇

```

Fragment referenced in 7ac, 13b, 43b.

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 43b > ≡
    SUCCES=0
    cd $modulesdir
    < move module (43c $DIR ) 6b >
    wget $URL
    SUCCES=$?
    if
        [ $SUCCES -eq 0 ]
    then
        tar -xzf $TARB
        SUCCES=$?
        rm -rf $TARB
    fi
    if
        [ $SUCCES -eq 0 ]
    then
        < logmess (43d Installed $DIR ) 43a >
        < remove old module (43e $DIR ) 6c >
    else
        < re-instate old module (43f $DIR ) 7a >
    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 T _E X 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 44 > ≡
  NUWEB=../env/bin/nuweb
  ◇
```

Fragment defined by 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Uses: nuweb 52b.

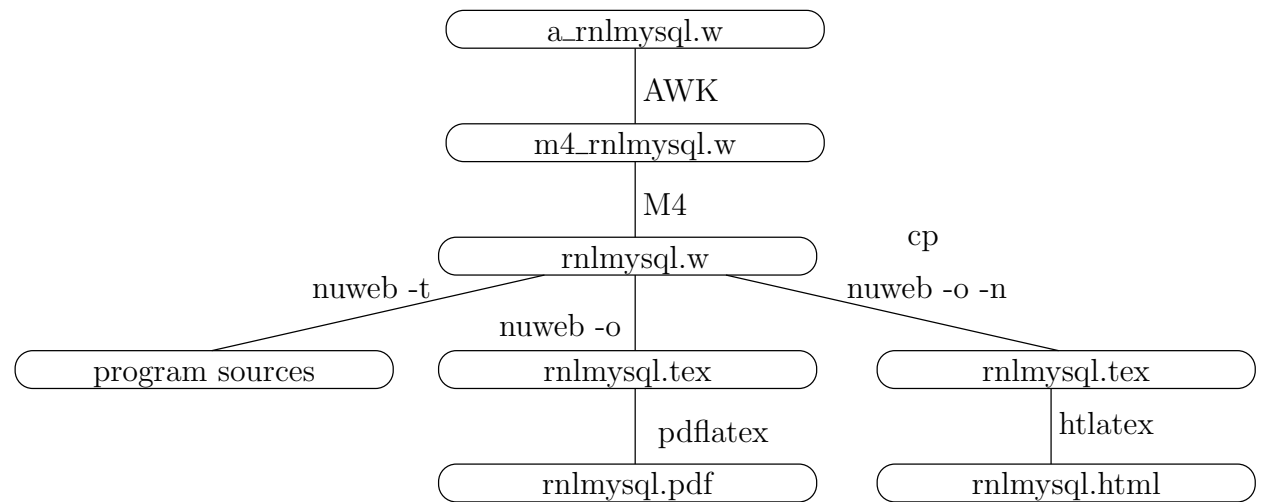


Figure 2: Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

```

"Makefile" 45a≡
  < default target 45b >

  < parameters in Makefile 44, ... >

  < impliciete make regels 49a, ... >
  < expliciete make regels 46c, ... >
  < make targets 45c, ... >
  ◇

```

The default target of make is `all`.

```

< default target 45b > ≡
  all : < all targets 46a >
  .PHONY : all

```

◇

Fragment referenced in 45a.
Defines: `all` 25a, `PHONY` 49b.

```

< make targets 45c > ≡
  clean:
    < clean up 9c, ... >

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.
Fragment referenced in 45a.

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

$\langle \text{all targets 46a} \rangle \equiv$
`nlpp.pdf`◇

Fragment referenced in 45b.

Uses: pdf 50a.

We use many suffixes that were not known by the C-programmers who constructed the `make` utility. Add these suffixes to the list.

$\langle \text{parameters in Makefile 46b} \rangle \equiv$
`.SUFFIXES: .pdf .w .tex .html .aux .log .php`

◇

Fragment defined by 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Defines: SUFFIXES Never used.

Uses: pdf 50a.

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.

$\langle \text{explicitete make regels 46c} \rangle \equiv$

```
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 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

Uses: nuweb 52b.

$\langle \text{clean up 46d} \rangle \equiv$
`rm -rf ../nuweb-1.58`

◇

Fragment defined by 9c, 10d, 19a, 37g, 46d.

Fragment referenced in 45c.

Uses: nuweb 52b.

```

⟨ expliciete make regels 47a ⟩ ≡
  ../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 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

Uses: nuweb 52b.

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 47b ⟩ ≡
  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 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

Uses: `print` 50a.

A.5.2 Run the M4 pre-processor

```

⟨ expliciete make regels 47c ⟩ ≡
  nlpp.w : m4_nlpp.w inst.m4
    m4 -P m4_nlpp.w > nlpp.w

```

◇

Fragment defined by 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

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 48a> ≡
    FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Defines: FIGFILES 48b.

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 48b> ≡
    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 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Defines: FIGFILENAMES Never used, PDFT_NAMES 50b, PDF_FIG_NAMES 50b, PST_NAMES Never used,
PS_FIG_NAMES Never used.

Uses: FIGFILES 48a.

Create the graph files with program `fig2dev`:


```

⟨ impliciete make regels 49a ⟩ ≡
    %.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 49a, 53c.

Fragment referenced in 45a.

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 49b ⟩ ≡
    bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

    .PHONY : bibfile

```

◇

Fragment defined by 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

Uses: PHONY 45b.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

```

< make targets 50a > ≡
    pdf : nlpp.pdf

    print : nlpp.pdf
           lpr nlpp.pdf

    view : nlpp.pdf
           evince nlpp.pdf

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.

Fragment referenced in 45a.

Defines: pdf 46ab, 50b, print 11c, 13a, 17b, 34a, 47b, view Never used.

Create the PDF document. This may involve multiple runs of nuweb, the L^AT_EX processor and the bibT_EX processor, and depends on the state of the aux file that the L^AT_EX processor creates as a by-product. Therefore, this is performed in a separate script, w2pdf.

The w2pdf script The three processors nuweb, L^AT_EX and bibT_EX are intertwined. L^AT_EX and bibT_EX 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 L^AT_EX 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 50b > ≡
    nlpp.pdf : nlpp.w $(W2PDF) $(PDF_FIG_NAMES) $(PDFT_NAMES)
              chmod 775 $(W2PDF)
              $(W2PDF) $*

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.

Fragment referenced in 45a.

Uses: pdf 50a, PDFT_NAMES 48b, PDF_FIG_NAMES 48b.

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 50c > ≡
    ../nuweb/bin ◇

```

Fragment defined by 4, 5abc, 9aef, 12d, 50c.

Fragment referenced in 56a.

Uses: nuweb 52b.

```

< parameters in Makefile 50d > ≡
    W2PDF=../nuweb/bin/w2pdf

```

◇

Fragment defined by 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Uses: nuweb 52b.

```

< expliciete make regels 51a > ≡
    $(W2PDF) : nlpp.w $(NUWEB)
              $(NUWEB) nlpp.w
◇

```

Fragment defined by 46c, 47abc, 49b, 51a, 53bd.
 Fragment referenced in 45a.

```

"../nuweb/bin/w2pdf" 51b≡
    #!/bin/bash
    # w2pdf -- compile a nuweb file
    # usage: w2pdf [filename]
    # 20151215 at 0908h: Generated by nuweb from a_nlpp.w
    NUWEB=../env/bin/nuweb
    LATEXCOMPILER=pdflatex
    < filenames in nuweb compile script 51d >
    < compile nuweb 51c >
◇

```

Uses: nuweb 52b.

The script retains a copy of the latest version of the auxiliary file. Then it runs the four processors nuweb, L^AT_EX, MakeIndex and bibT_EX, until they do not change the auxiliary file or the index.

```

< compile nuweb 51c > ≡
    NUWEB=/home/huygen/projecten/pipelines/nlpp/env/bin/nuweb
    < run the processors until the aux file remains unchanged 52c >
    < remove the copy of the aux file 52a >
◇

```

Fragment referenced in 51b.
 Uses: nuweb 52b.

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 L^AT_EX 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 51d > ≡
    nufil=$1
    trunk=${1%.*}
    texfil=${trunk}.tex
    auxfil=${trunk}.aux
    oldaux=old.${trunk}.aux
    indexfil=${trunk}.idx
    oldindexfil=old.${trunk}.idx
◇

```

Fragment referenced in 51b.

Defines: auxfil 52c, 54c, 55a, indexfil 52c, 54c, nufil 52b, 54c, 55b, oldaux 52ac, 54c, 55a, oldindexfil 52c, 54c, texfil 52b, 54c, 55b, trunk 52b, 54c, 55bc.

Remove the old copy if it is no longer needed.

```

⟨ remove the copy of the aux file 52a ⟩ ≡
    rm $oldaux
    ◇

```

Fragment referenced in 51c, 54b.
 Uses: oldaux 51d, 54c.

Run the three processors. Do not use the option `-o` (to suppress generation of program sources) for nuweb, because `w2pdf` must be kept up to date as well.

```

⟨ run the three processors 52b ⟩ ≡
    $NUWEB $nufil
    $LATEXCOMPILER $texfil
    makeindex $trunk
    bibtex $trunk
    ◇

```

Fragment referenced in 52c.
 Defines: bibtex 55bc, makeindex 55bc, nuweb 5d, 42a, 44, 46cd, 47a, 50cd, 51bc, 53a, 54a.
 Uses: nufil 51d, 54c, texfil 51d, 54c, trunk 51d, 54c.

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 52c ⟩ ≡
    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 52b ⟩
        if [ $LOOPCOUNTER -ge 10 ]
        then
            cp $auxfil $oldaux
        fi;
    done
    ◇

```

Fragment referenced in 51c.
 Uses: auxfil 51d, 54c, indexfil 51d, oldaux 51d, 54c, oldindexfil 51d.

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 53a> ≡
    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 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Uses: nuweb 52b.

Make the directory:

```
<expliciete make regels 53b> ≡
    $(htmldir) :
        mkdir -p $(htmldir)
◇
```

Fragment defined by 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

The rule to copy files in it:

```
<impliciete make regels 53c> ≡
    $(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
◇
```

Fragment defined by 49a, 53c.

Fragment referenced in 45a.

Do the work:

```
<expliciete make regels 53d> ≡
    $(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
◇
```

Fragment defined by 46c, 47abc, 49b, 51a, 53bd.

Fragment referenced in 45a.

Invoke:

```
<make targets 53e> ≡
    htm : $(htmldir) $(htmltarget)
◇
```

Fragment defined by 45c, 50ab, 53e, 56acd.

Fragment referenced in 45a.

Create a script that performs the translation.

```
"w2html" 54a≡
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
# [filename]: Name of the nuweb source file.
# 20151215 at 0908h: Generated by nuweb from a_nlpp.w
echo "translate " $1 >w2html.log
NUWEB=/home/huygen/projecten/pipelines/nlpp/env/bin/nuweb
⟨filenames in w2html 54c⟩

⟨perform the task of w2html 54b⟩
```

◇

Uses: **nuweb** 52b.

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 54b⟩ ≡
  ⟨run the html processors until the aux file remains unchanged 55a⟩
  ⟨remove the copy of the aux file 52a⟩
◇
```

Fragment referenced in 54a.

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 L^AT_EX 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 54c⟩ ≡
nufil=$1
trunk=${1%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```

Fragment referenced in 54a.

Defines: **auxfil** 51d, 52c, 55a, **nufil** 51d, 52b, 55b, **oldaux** 51d, 52ac, 55a, **texfil** 51d, 52b, 55b, **trunk** 51d, 52b, 55bc.

Uses: **indexfil** 51d, **oldindexfil** 51d.

```

⟨run the html processors until the aux file remains unchanged 55a⟩ ≡
    while
        ! cmp -s $auxfil $oldaux
    do
        if [ -e $auxfil ]
        then
            cp $auxfil $oldaux
        fi
        ⟨run the html processors 55b⟩
    done
    ⟨run tex4ht 55c⟩

```

◇

Fragment referenced in 54b.

Uses: auxfil 51d, 54c, oldaux 51d, 54c.

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

```

⟨run the html processors 55b⟩ ≡
    $NUWEB -o -n $nufil
    latex $texfil
    makeindex $trunk
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 55a.

Uses: bibtex 52b, makeindex 52b, nufil 51d, 54c, texfil 51d, 54c, trunk 51d, 54c.

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 55c⟩ ≡
    tex '\def\filename{{\nlpp}{\idx}{4dx}{\ind}} \input idxmake.4ht'
    makeindex -o $trunk.ind $trunk.4dx
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 55a.

Uses: bibtex 52b, makeindex 52b, trunk 51d, 54c.

A.7 Create the program sources

Run nuweb, but suppress the creation of the L^AT_EX 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 55d⟩ ≡
    MKDIR = mkdir -p

```

◇

Fragment defined by 44, 46b, 48ab, 50d, 53a, 55d.

Fragment referenced in 45a.

Defines: MKDIR 56a.

```

< make targets 56a > ≡
    DIRS = < directories to create 4, ... >

```

```

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

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.
 Fragment referenced in 45a.
 Defines: DIRS 56c.
 Uses: MKDIR 55d.

```

< make scripts executable 56b > ≡
    chmod -R 775 ../bin/*
    chmod -R 775 ../env/bin/*

```

◇

Fragment defined by 16b, 17d, 56b.
 Fragment referenced in 56c.

```

< make targets 56c > ≡
    sources : nlpp.w $(DIRS) $(NUWEB)
              $(NUWEB) nlpp.w
              < make scripts executable 16b, ... >

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.
 Fragment referenced in 45a.
 Uses: DIRS 56a.

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 56d > ≡
    transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant

```

◇

Fragment defined by 45c, 50ab, 53e, 56acd.
 Fragment referenced in 45a.

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


```

"../env/bin/transplant" 57≡
    #!/bin/bash
    LOGLEVEL=1
    < set variables that point to the directory-structure 5d, ... >
    < set paths after transplantation 13a >
    < re-install modules after the transplantation 21d >

◇

```

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/coreference-base" Defined by 27c.
"../bin/dbpner" Defined by 39b.
"../bin/evcoref" Defined by 38c.
"../bin/framesrl" Defined by 34a.
"../bin/heideltime" Defined by 36b.
"../bin/install-modules" Defined by 15, 16a.
"../bin/langdetect" Defined by 17c.
"../bin/lu2synset" Defined by 31c.
"../bin/mor" Defined by 26d.
"../bin/ned" Defined by 32b.
"../bin/nerc_conll02" Defined by 29b.
"../bin/nomevent" Defined by 39d.
"../bin/onto" Defined by 33.
"../bin/opinimin" Defined by 41.
"../bin/postersrl" Defined by 38a.
"../bin/srl" Defined by 36d, 37abcd.
"../bin/start-spotlight" Defined by 23a.
"../bin/test" Defined by 42a.
"../bin/tok" Defined by 26b.
"../bin/wsd" Defined by 30c.
"../env/bin/langdetect.py" Defined by 17b.
"../env/bin/progenv" Defined by 6a, 8d.
"../env/bin/transplant" Defined by 57.
"../env/etc/opini_nl.cfg" Defined by 40b.
"../nuweb/bin/w2pdf" Defined by 51b.
"Makefile" Defined by 45a.
"w2html" Defined by 54a.

```

C.2 Macro's

```

< abort when the language is not English or Dutch 18b > Referenced in 26b.
< activate the install-to-project-repo utility 35d > Referenced in 35b.

```

< activate the python environment 12ce > Referenced in 11ab.
 < all targets 46a > Referenced in 45b.
 < check this first 8a, 16c > Referenced in 15.
 < check whether mercurial is present 17a > Referenced in 16c.
 < check/install the correct version of python 11c > Referenced in 11b.
 < clean up 9c, 10d, 19a, 37g, 46d > Referenced in 45c.
 < clone the heideltime wrapper 34c > Referenced in 34b.
 < compile nuweb 51c > Referenced in 51b.
 < compile the heideltime wrapper 35b > Referenced in 34b.
 < compile the nerc jar 28b > Referenced in 28a.
 < create a virtual environment for Python 12a > Referenced in 11b.
 < create javapython script 8c > Referenced in 15.
 < default target 45b > Referenced in 45a.
 < directories to create 4, 5abc, 9aef, 12d, 50c > Referenced in 56a.
 < download svm models 30b > Referenced in 29c.
 < expliciete make regels 46c, 47abc, 49b, 51a, 53bd > Referenced in 45a.
 < filenames in nuweb compile script 51d > Referenced in 51b.
 < filenames in w2html 54c > Referenced in 54a.
 < get jvntextpro-2.0.jar 35c > Referenced in 35b.
 < get or have ? > Referenced in 31a, 37e.
 < get the mor time-out parameter 27a > Referenced in 26d.
 < get the nerc models 29a > Referenced in 28a.
 < get the snapshot 8b > Referenced in 15.
 < impliciete make regels 49a, 53c > Referenced in 45a.
 < install ActivePython 11d > Referenced in 11c.
 < install Alpino 18c > Referenced in 15.
 < install coreference-base 27b > Referenced in 16a.
 < install CRFsuite 25b > Referenced in 15.
 < install from github 7c > Referenced in 26c, 27b, 29c, 31d, 34c, 36c, 39a.
 < install from tarball 43b > Not referenced.
 < install kafnaparserpy 13b > Referenced in 11b.
 < install maven 10bc > Referenced in 10a.
 < install maven if it hasn't been done 10a > Referenced in 15.
 < install python if it hasn't been done 11a > Not referenced.
 < install python packages 14a > Referenced in 11b.
 < install svm lib 30a > Referenced in 29c.
 < install SVMLight 25a > Referenced in 15.
 < install the dbpedia-ner module 39a > Referenced in 16a.
 < install the event-coreference module 38b > Referenced in 16a.
 < install the heideltime module 34b > Referenced in 16a.
 < install the lu2synset converter 31a > Referenced in 16a.
 < install the morphosyntactic parser 26c > Referenced in 16a.
 < install the NERC module 28a > Referenced in 16a.
 < install the nomevent module 39c > Referenced in 16a.
 < install the onto module 32c > Referenced in 16a.
 < install the opinion-miner 40ac > Referenced in 16a.
 < install the post-SRL module 37e > Referenced in 16a.
 < install the Spotlight server 22bc > Referenced in 22a.
 < install the Spotlight server if it hasn't been done 22a > Referenced in 15.
 < install the srl module 36c > Referenced in 16a.
 < install the ticcutils utility 21a > Referenced in 15, 21d.
 < install the timbl utility 21b > Referenced in 15, 21d.
 < install the tokenizer 26a > Referenced in 16a.
 < install the treetagger if it hasn't been done 19b > Referenced in 15.
 < install the treetagger utility 19cde, 20abcd > Referenced in 19b.
 < install the WSD module 29c > Referenced in 16a.
 < install the NED module 31d > Referenced in 16a.
 < install VUA-pylib 24c > Referenced in 15.

<logmess [43a](#)> Referenced in [7ac](#), [13b](#), [43b](#).
 <make scripts executable [16b](#), [17d](#), [56b](#)> Referenced in [56c](#).
 <make targets [45c](#), [50ab](#), [53e](#), [56acd](#)> Referenced in [45a](#).
 <move module [6b](#)> Referenced in [7c](#), [13b](#), [43b](#).
 <parameters in Makefile [44](#), [46b](#), [48ab](#), [50d](#), [53a](#), [55d](#)> Referenced in [45a](#).
 <perform the task of w2html [54b](#)> Referenced in [54a](#).
 <put Antske's material in the heideltime wrapper [35a](#)> Referenced in [34b](#).
 <put spotlight jar in the Maven repository [32a](#)> Referenced in [31d](#).
 <re-install modules after the transplantation [21d](#)> Referenced in [57](#).
 <re-instate old module [7a](#)> Referenced in [7c](#), [13b](#), [43b](#).
 <read the list of installed modules [14b](#)> Referenced in [15](#).
 <remove old module [6c](#)> Referenced in [7c](#), [13b](#), [43b](#).
 <remove outdated heideltime jars [36a](#)> Referenced in [35d](#).
 <remove the copy of the aux file [52a](#)> Referenced in [51c](#), [54b](#).
 <run tex4ht [55c](#)> Referenced in [55a](#).
 <run the html processors [55b](#)> Referenced in [55a](#).
 <run the html processors until the aux file remains unchanged [55a](#)> Referenced in [54b](#).
 <run the processors until the aux file remains unchanged [52c](#)> Referenced in [51c](#).
 <run the three processors [52b](#)> Referenced in [52c](#).
 <set alpinohome [18d](#)> Referenced in [26d](#).
 <set paths after transplantation [13a](#)> Referenced in [57](#).
 <set the language variable [18a](#)> Referenced in [42a](#).
 <set up java [9bd](#)> Referenced in [15](#).
 <set up python [11b](#)> Referenced in [11a](#), [15](#).
 <set variables that point to the directory-structure [5de](#), [7h](#)> Referenced in [6a](#), [15](#), [57](#).
 <start the Spotlight server on localhost [24b](#)> Referenced in [23b](#).
 <test whether spotlighthost runs [24a](#)> Referenced in [23b](#).
 <test whether virtualenv is present on the host [12b](#)> Referenced in [12a](#).
 <try to obtain a running spotlightserver [23b](#)> Referenced in [23a](#).
 <unpack ticcutils or timbl [21c](#)> Referenced in [21ab](#).
 <variables of install-modules [42c](#)> Referenced in [15](#).

C.3 Variables

activate: [12c](#), [13a](#).
 all: [25a](#), [45b](#).
 ALPINO_HOME: [18d](#).
 auxfil: [51d](#), [52c](#), [54c](#), [55a](#).
 bibtex: [52b](#), [55bc](#).
 DIRS: [56a](#), [56c](#).
 fig2dev: [49a](#).
 FIGFILENAMES: [48b](#).
 FIGFILES: [48a](#), [48b](#).
 hg: [17a](#), [27b](#).
 indexfil: [51d](#), [52c](#), [54c](#).
 lang: [17b](#), [18a](#).
 lxml: [14a](#).
 makeindex: [52b](#), [55bc](#).
 MKDIR: [55d](#), [56a](#).
 nufil: [51d](#), [52b](#), [54c](#), [55b](#).
 nuweb: [5d](#), [42a](#), [44](#), [46cd](#), [47a](#), [50cd](#), [51bc](#), [52b](#), [53a](#), [54a](#).
 oldaux: [51d](#), [52ac](#), [54c](#), [55a](#).
 oldindexfil: [51d](#), [52c](#), [54c](#).
 PATH: [5e](#), [9d](#), [10c](#), [41](#).
 pdf: [46ab](#), [50a](#), [50b](#).
 PDFT_NAMES: [48b](#), [50b](#).
 PDF_FIG_NAMES: [48b](#), [50b](#).
 PHONY: [45b](#), [49b](#).

`print`: [11c](#), [13a](#), [17b](#), [34a](#), [47b](#), [50a](#).
`PST_NAMES`: [48b](#).
`PS_FIG_NAMES`: [48b](#).
`pythonok`: [11c](#).
`PYTHONPATH`: [12e](#).
`pyyaml`: [14a](#).
`SUFFIXES`: [46b](#).
`texfil`: [51d](#), [52b](#), [54c](#), [55b](#).
`trunk`: [51d](#), [52b](#), [54c](#), [55bc](#).
`view`: [50a](#).
`virtualenv`: [11d](#), [12a](#), [12b](#).