Standardised Dutch NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

8th July 2015
09:57 h.

Abstract

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

Contents

1 Introduction

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

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

The installation has been parameterised. The locations and names that you read (and that will be used to build the pipeline) have been read from variables in file `inst.m4` in the nuweb directory.

1. http://wordpress.let.vupr.nl
2. http://www.newsreader-project.eu
3. https://surfsara.nl/systems/lisa

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	snapshot	56f83ce4b61680346f15e5d4e6de6293764f7383	`tok`
morphosyntactic parser	4.4.2	Github	c6cabea2cc37ac3098c5927f5ec5b180ac31246f	`mor`
NERC	4.4.4	Github	9927fdb32d943f0aa9748a656958af99eeb1f5b7	nerc
WSD	4.4.5	Github	2babeb40a81b3720274a0521ccc2a27c5eff28c9	wsd
Onto-tagger	4.4.8	snapshot		onto
Heideltime	4.4.10	Gith./snap.	057c93ccc857a427145b9e2ff72fd645172d34df	heideltime
SRL	4.4.11	Github	675d22d361289ede23df11dcdb17195f008c54bf	srl
NED	4.4.7	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	4.4.3	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	4.4.13	snapshot		evcoref
Framenet SRL	4.4.9	snapshot		fsrl
Dbpedia_ner	4.4.14	Github	ab1dcbd860f0ff29bc979f646dc382122a101fc2	dbpner

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

The modules are obtained in one of the following ways:

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

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

Module	Version	Section	Source
KafNafParserPy	Feb 1, 2015	3.3.2	Github
Alpino	20706	4.3.1	RUG
Ticcutils	0.7	4.3.3	ILK
Timbl	6.4.6	4.3.3	ILK
Treetagger	3.2	4.3.2	Uni. München
Spotlight server	0.7	4.3.4	Spotlight

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

1.2 File-structure of the pipeline

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

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.

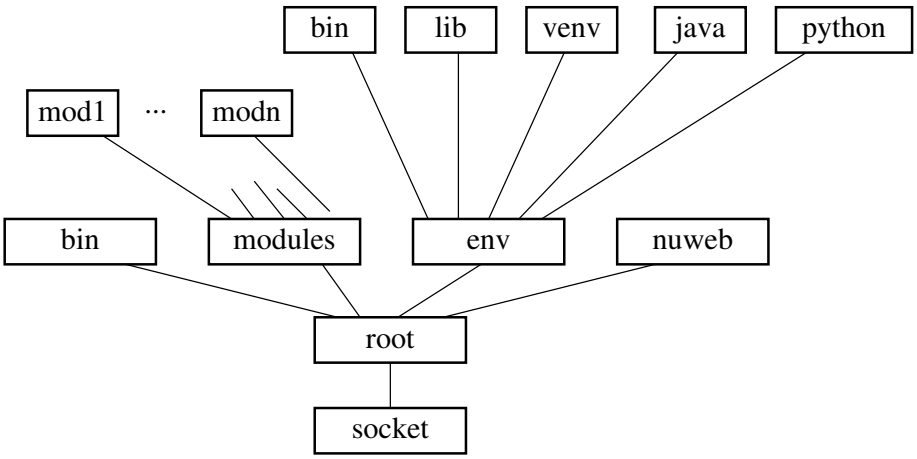

Figure 1: *Directory-structure of the pipeline (see text).*

env: The programming environment. It contains a.o. the Java development kit, Python, the Python virtual environment (`venv`), libraries and binaries.

⟨ *directories to create* 4a ⟩ ≡
```
../modules ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

⟨ *directories to create* 4b ⟩ ≡
```
../bin ../env/bin ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

⟨ *directories to create* 4c ⟩ ≡
```
../env/lib ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

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

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

```
if
  [ "$piperoot" == "" ]
then
  export piperoot=/mnt/sdb1/pipelines/testnlpp/nlpp
fi
export pipesocket=${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 5abc.
Fragment referenced in 13, 20c, 21bd, 23cd, 24e, 26b, 27, 28a, 29d, 30b, 31d, 32ac, 33c, 47b.
Uses: `nuweb` 42d.

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

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

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

Fragment defined by 5abc.
Fragment referenced in 13, 20c, 21bd, 23cd, 24e, 26b, 27, 28a, 29d, 30b, 31d, 32ac, 33c, 47b.
Defines: `PATH` 9d, 10a.

While setting variables, *source* a scripts that sets variables for directories of which we do not yet know where they are, e.g. paths to Python and Java that we may have to set up dynamically.

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

```
source $envbindir/progenv
◇
```

Fragment defined by 5abc.
Fragment referenced in 13, 20c, 21bd, 23cd, 24e, 26b, 27, 28a, 29d, 30b, 31d, 32ac, 33c, 47b.

2 How to obtain modules and other material

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

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

2.1 Location-dependency

The basic way of installation is, to clone this repository from Github on the intended location in the file-system of the target computer and then run the install-scripts. However, it may be advantageous to be able to transplant a complete installation to another location in another computer. This could be done by making all path-descriptions in all scripts relative to anchorpoints

within the installation, while it may be hard to find such anchorpoints in advance. Therefore, we take another approach in which we supply a script that repairs paths-descriptions after the transplantation (section A.8).

2.2 Reversible update

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

⟨ *move module* 6a ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
```
◇

Fragment referenced in 7a, 12b, 34c.

⟨ *remove old module* 6b ⟩ ≡
```
rm -rf old.@1
```
◇

Fragment referenced in 7a, 12b, 34c.

⟨ *re-instate old module* 6c ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
```
⟨ *logmess* (6d `$MESS`) 34b ⟩

◇

Fragment referenced in 7a, 12b, 34c.

2.3 Installation from Github

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

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

⟨ *install from github* 7a ⟩ ≡

```
cd $modulesdir
⟨ move module (7b $DIRN ) 6a ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (7c Cannot install current $MODNAM version ) 34b ⟩
  ⟨ re-instate old module (7d $DIRN ) 6c ⟩
else
  ⟨ remove old module (7e $DIRN ) 6b ⟩
  cd $modulesdir/$DIRN
  git checkout $GITC
fi

◇
```

Fragment referenced in 21ac, 24a, 25d, 28b, 30a, 32b.

2.4 Installation from the snapshot

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

⟨ *check this first* 7f ⟩ ≡

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

Fragment defined by 7f, 14c.
Fragment referenced in 13.

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

⟨ *get or have* 8a ⟩ ≡

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

◇

Fragment referenced in 9a, 11a, 15b, 19a, 23a, 24c, 25a, 26c, 31ae, 33a.

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 progenv script* 8b ⟩ ≡

```
echo '#!/bin/bash' > /mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/progenv
```

◇

Fragment referenced in 13.

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* 8c ⟩ ≡

```
../env/java ◇
```

Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

⟨ *set up java* 9a ⟩ ≡
```
⟨ get or have (9b server-jre-7u72-linux-x64.tar.gz ) 8a ⟩
cd $envdir/java
tar -xzf $pipesocket/server-jre-7u72-linux-x64.tar.gz
◇
```
Fragment defined by 9ad.
Fragment referenced in 13.

Remove the java-ball when cleaning up:

⟨ *clean up* 9c ⟩ ≡
```
rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz
◇
```
Fragment defined by 9c, 10b, 15f, 31c, 37c.
Fragment referenced in 36d.

⟨ *set up java* 9d ⟩ ≡
```
echo 'export JAVA_HOME=$envdir/java/jdk1.7.0_72' >> /mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/progenv
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/progenv
export JAVA_HOME=$envdir/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 9ad.
Fragment referenced in 13.
Uses: PATH 5b.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 9e ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

3.2 Maven

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

⟨ *directories to create* 9f ⟩ ≡
```
../env/apache-maven-3.0.5 ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

⟨ *install maven* 9g ⟩ ≡
```
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 9g, 10a.
Fragment referenced in 13.

⟨ *install maven* 10a ⟩ ≡
```
export MAVEN_HOME=$envdir/apache-maven-3.0.5
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```
Fragment defined by 9g, 10a.
Fragment referenced in 13.
Uses: PATH 5b.

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

⟨ *clean up* 10b ⟩ ≡
```
rm -rf ../env/apache-maven-3.0.5
◇
```
Fragment defined by 9c, 10b, 15f, 31c, 37c.
Fragment referenced in 36d.

3.3 Python

Set up the environment for Python (version 2.7). I could not find an easy way to set up Python from scratch. Therefore we wil use Python 2.7 if is has been installed on the host. Otherwise, we will use a binary distribution obtained from ActiveState. A tarball of ActivePython can be obtained from the snapshot.

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

⟨ *set up python* 10c ⟩ ≡
```
⟨ check/install the correct version of python 10d ⟩
⟨ create a virtual environment for Python 11c ⟩
⟨ activate the python environment 11e, ... ⟩
⟨ install kafnafparserpy 12b ⟩
⟨ install python packages 12g ⟩
◇
```
Fragment referenced in 13.

⟨ *check/install the correct version of python* 10d ⟩ ≡
```
pythonok=‘python --
version 2>&1 | gawk ’{if(match($2, "2.7")) print "yes" ; else print "no" }’‘
if
  [ "$pythonok" == "no" ]
then
  ⟨ install ActivePython 11a ⟩
fi
◇
```
Fragment referenced in 10c.
Defines: pythonok Never used.
Uses: print 40b.

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* 11a ⟩ ≡
```
⟨ get or have (11b ActivePython-2.7.8.10-linux-x86_64.tar.gz ) 8a ⟩
pytinsdir=‘mktemp -d -t activepyt.XXXXXX‘
cd $pytinsdir
tar -xzf $pipesocket/ActivePython-2.7.8.10-linux-x86_64.tar.gz
acdir=‘ls -1‘
cd $acdir
./install.sh -I $envdir
cd $piperoot
rm -rf $pytinsdir
pip install -U pip virtualenv setuptools
◇
```
Fragment referenced in 10d.

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* 11c ⟩ ≡
```
⟨ test whether virtualenv is present on the host 11d ⟩
cd $envdir
virtualenv venv
◇
```
Fragment referenced in 10c.
Uses: virtualenv 11d.

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

⟨ *activate the python environment* 11e ⟩ ≡
```
source $envdir/venv/bin/activate
echo 'source $en-
vdir/venv/bin/activate' >> /mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/progenv
◇
```
Fragment defined by 11e, 12a.
Fragment referenced in 10c.
Defines: activate Never used.

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

⟨ *directories to create* 11f ⟩ ≡
```
../env/python ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.

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

⟨ *activate the python environment* 12a ⟩ ≡
```
echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/progenv
export PYTHONPATH=$envdir/python:$PYTHONPATH
```
◇

Fragment defined by 11e, 12a.
Fragment referenced in 10c.
Defines: `PYTHONPATH` Never used.

3.3.2 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* 12b ⟩ ≡
```
cd $envdir/python
DIRN=KafNafParserPy
⟨ move module (12c $DIRN ) 6a ⟩
git clone https://github.com/cltl/KafNafParserPy.git
if
  [ $? -gt 0 ]
then
  ⟨ logmess (12d Cannot install current $DIRN version ) 34b ⟩
  ⟨ re-instate old module (12e $DIRN ) 6c ⟩
else
  ⟨ remove old module (12f $DIRN ) 6b ⟩
fi
```
◇

Fragment referenced in 10c.

3.3.3 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* 12g ⟩ ≡
```
pip install lxml
pip install pyyaml
```
◇

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

4 Installation of the modules

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

4.1 The installation script

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

```
"../bin/install-modules" 13≡
    #!/bin/bash
    echo Set up environment
    ⟨ create progenv script 8b ⟩
    ⟨ set variables that point to the directory-structure 5a, ... ⟩
    ⟨ variables of install-modules 34a ⟩
    ⟨ check this first 7f, ... ⟩
    echo ... Java
    ⟨ set up java 9a, ... ⟩
    ⟨ install maven 9g, ... ⟩
    echo ... Python
    ⟨ set up python 10c ⟩
    echo ... Alpino
    ⟨ install Alpino 15b ⟩
    echo ... Spotlight
    ⟨ install the Spotlight server 19a, ... ⟩
    echo ... Treetagger
    ⟨ install the treetagger utility 16a, ... ⟩
    echo ... Ticcutils and Timbl
    ⟨ install the ticcutils utility 17c ⟩
    ⟨ install the timbl utility 17d ⟩
    ◇
```

File defined by 13, 14a.

Next, install the modules:

```
"../bin/install-modules" 14a≡
        echo Install modules
        echo ... Tokenizer
        ⟨ install the tokenizer 20b ⟩
        echo ... Morphosyntactic parser
        ⟨ install the morphosyntactic parser 21a ⟩
        echo ... NERC
        ⟨ install the NERC module 22a ⟩
        echo ... Coreference base
        ⟨ install coreference-base 21c ⟩
        echo ... WSD
        ⟨ install the WSD module 24a ⟩
        echo ... Ontotagger
        ⟨ install the onto module 26c ⟩
        echo ... Heideltime
        ⟨ install the heideltime module 28b ⟩
        echo ... SRL
        ⟨ install the srl module 30a ⟩
        echo ... NED
        ⟨ install the NED module 25d ⟩
        echo ... Event-coreference
        ⟨ install the event-coreference module 31e ⟩
        echo ... lu2synset
        ⟨ install the lu2synset converter 25a ⟩
        echo ... dbpedia-ner
        ⟨ install the dbpedia-ner module 32b ⟩
        echo ... nominal event
        ⟨ install the nomevent module 33a ⟩
        ⟨ install the post-SRL module 31a ⟩
        echo Final
        ◇
```

File defined by 13, 14a.

```
⟨ make scripts executable 14b ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```

Fragment defined by 14b, 46c.
Fragment referenced in 46d.

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 14c ⟩ ≡
        ⟨ check whether mercurial is present 15a ⟩
        ◇
```

Fragment defined by 7f, 14c.
Fragment referenced in 13.

⟨ *check whether mercurial is present* 15a ⟩ ≡
```
which hg
if
  [ $? -ne 0 ]
then
  echo Please install Mercurial.
  exit 1
fi
```
◇

Fragment referenced in 14c.
Defines: `hg` 21c.

4.3 Install utilities and resources

4.3.1 Alpino

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

Module

⟨ *install Alpino* 15b ⟩ ≡
```
⟨ get or have (15c Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
⟨ logmess (15d Installed Alpino ) 34b ⟩
```
◇

Fragment referenced in 13.

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* 15e ⟩ ≡
```
export ALPINO_HOME=$modulesdir/Alpino
```
◇

Fragment referenced in 21b.
Defines: `ALPINO_HOME` Never used.

Remove the tarball when cleaning up:

⟨ *clean up* 15f ⟩ ≡
```
rm -rf $pipesocket/Alpino-x86_64-linux-glibc2.5-20706-sicstus.tar.gz
```
◇

Fragment defined by 9c, 10b, 15f, 31c, 37c.
Fragment referenced in 36d.

4.3.2 Treetagger

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

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

2. Download and unpack the tagger-scripts tarball

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

⟨ *install the treetagger utility* 16a ⟩ ≡

```
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 16abcde, 17ab.
Fragment referenced in 13.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 16b ⟩ ≡

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

Fragment defined by 16abcde, 17ab.
Fragment referenced in 13.

Parametersets:

⟨ *install the treetagger utility* 16c ⟩ ≡

```
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 16abcde, 17ab.
Fragment referenced in 13.

Download everything in the target directory:

⟨ *install the treetagger utility* 16d ⟩ ≡

```
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 16abcde, 17ab.
Fragment referenced in 13.

Run the install-script:

⟨ *install the treetagger utility* 16e ⟩ ≡

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

Fragment defined by 16abcde, 17ab.
Fragment referenced in 13.

Make the treetagger utilities available for everybody.

⟨ *install the treetagger utility* 17a ⟩ ≡
```
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 16abcde, 17ab.
Fragment referenced in 13.

Remove the tarballs:

⟨ *install the treetagger utility* 17b ⟩ ≡
```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```
Fragment defined by 16abcde, 17ab.
Fragment referenced in 13.

4.3.3 Timbl and Ticcutils

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

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

⟨ *install the ticcutils utility* 17c ⟩ ≡
```
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 18a ⟩
◇
```
Fragment referenced in 13, 18d.

⟨ *install the timbl utility* 17d ⟩ ≡
```
URL=http://software.ticc.uvt.nl/timbl-6.4.6.tar.gz
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
⟨ unpack ticcutils or timbl 18a ⟩
◇
```
Fragment referenced in 13, 18d.

⟨ *unpack ticcutils or timbl* 18a ⟩ ≡

```
SUCCES=0
ticbeldir=‘mktemp -t -d tickbel.XXXXXX‘
cd $ticbeldir
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  cd $DIR
  ./configure --prefix=$envdir
  make
  make install
fi
cd $piperoot
rm -rf $ticbeldir
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (18b Installed $DIR ) 34b ⟩
else
  ⟨ logmess (18c NOT installed $DIR ) 34b ⟩
fi
◇
```

Fragment referenced in 17cd.

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

⟨ *re-install modules after the transplantation* 18d ⟩ ≡

```
⟨ install the ticcutils utility 17c ⟩
⟨ install the timbl utility 17d ⟩
◇
```

Fragment referenced in 47b.

4.3.4 Spotlight

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

> The NED module works for English, Spanish, Dutch and Italian. The module returns multiple candidates and correspondences for all the languages. If you want to integrate it in your Dutch or Italian pipeline, you will need:
>
> 1. The jar file with the dbpedia-spotlight server. You need the version that Aitor developed in order to correctly use the ”candidates” option. You can copy it from the English VM. The jar file name is `dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar`
> 2. The Dutch/Italian model for the dbpedia-spotlight. You can download them from: http://spotlight.sztaki.hu/downloads/

3. The jar file with the NED module: `ixa-pipe-ned-1.0.jar`. You can copy it from the English VM too.
4. The file: `wikipedia-db.v1.tar.gz`. You can download it from: http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz. This file contains the required information to do the mappings between the wikipedia-entries. The zip file contains three files: wikipedia-db, wikipedia-db.p and wikipedia-db.t

To start the dbpedia server: Italian server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar \
    it http://localhost:2050/rest
```

Dutch server:

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

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

So, let's do that.

⟨ *install the Spotlight server* 19a ⟩ ≡
⟨ *get or have* (19b `spotlightnl.tgz`) 8a ⟩
```
cd $envdir
tar -xzf $pipesocket/spotlightnl.tgz
cd $envdir/spotlight
wget http://spotlight.sztaki.hu/downloads/nl.tar.gz
tar -xzf nl.tar.gz
rm nl.tar.gz
```
◇
Fragment defined by 19ac.
Fragment referenced in 13.

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

⟨ *install the Spotlight server* 19c ⟩ ≡
```
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 19ac.
Fragment referenced in 13.

⟨ *start the Spotlight server* 19d ⟩ ≡
```
cd /mnt/sdb1/pipelines/testnlpp/nlpp/env/spotlight
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-
candidates.jar nl http://localhost:2060/rest  &
```
◇
Fragment referenced in 20a.

We start the spotlight-server only in case it is not already running. Assume that Spotlight runs when something listens on port 2060 of localhost:

⟨ *check/start the Spotlight server* 20a ⟩ ≡

```
spottasks=‘netstat -an | grep :2060 | wc -l‘
if
  [ $spottasks -eq 0 ]
then
  ⟨ start the Spotlight server 19d ⟩
  sleep 60
fi
◇
```

Fragment referenced in 26b, 32c.

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* 20b ⟩ ≡

```
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 14a.

Script The script runs the tokenizerscript.

"../bin/tok" 20c≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, . . . ⟩
JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
java -Xmx1000m  -jar $JARFILE tok -l nl --inputkaf
◇
```

4.4.2 Morphosyntactic parser

Module

⟨ *install the morphosyntactic parser* 21a ⟩ ≡

```
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=c6cabea2cc37ac3098c5927f5ec5b180ac31246f
⟨ install from github 7a ⟩
cd $modulesdir/morphosyntactic_parser_nl
git checkout c6cabea2cc37ac3098c5927f5ec5b180ac31246f
◇
```

Fragment referenced in 14a.

Script

"../bin/mor" 21b≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
ROOT=$piperoot
MODDIR=$modulesdir/morphosyntactic_parser_nl
⟨ set alpinohome 15e ⟩
cat | python $MODDIR/core/morph_syn_parser.py
◇
```

4.4.3 Nominal coreference-base

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

Module

⟨ *install coreference-base* 21c ⟩ ≡

```
MODNAM=coreference-base
DIRN=coreference-base
GITU=https://github.com/opener-project/coreference-base.git
GITC=bfa5aec0fa498e57fe14dd4d2c51365dd09a0757
⟨ install from github 7a ⟩
pip install --upgrade  hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
pip install --upgrade  networkx
◇
```

Fragment referenced in 14a.
Uses: hg 15a.

Script

"../bin/coreference-base" 21d≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
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* 22a ⟩ ≡
⟨ *compile the nerc jar* 22b ⟩
⟨ *get the nerc models* 23a ⟩

◇

Fragment referenced in 14a.

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

⟨ *compile the nerc jar* 22b ⟩ ≡

```
TEMPDIR==‘mktemp -d -t nerc.XXXXXX‘
cd $TEMPDIR
git clone https://github.com/ixa-ehu/ixa-pipe-nerc
cd ixa-pipe-nerc/
git checkout 9927fdb32d943f0aa9748a656958af99eeb1f5b7
mvn clean package
mv target/ixa-pipe-nerc-1.3.6.jar $jarsdir/
cd $nuwebdir
rm -rf $TEMPDIR
```

◇

Fragment referenced in 22a.

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

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

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

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

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

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

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

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

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

⟨ *get the nerc models* 23a ⟩ ≡
```
⟨ get or have (23b EHU-nerc.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/EHU-nerc.tgz
chmod -R 775 $modulesdir/EHU-nerc
◇
```
Fragment referenced in 22a.

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

"../bin/nerc_conll02" 23c≡
```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
MODDIR=$modulesdir/EHU-nerc
JAR=$jarsdir/ixa-pipe-nerc-1.3.6.jar
MODEL=nl-clusters-conll02.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL
◇
```

"../bin/nerc_sonar" 23d≡
```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
MODDIR=$modulesdir/EHU-nerc
JAR=$jarsdir/ixa-pipe-nerc-1.3.6.jar
MODEL=nl-clusters-sonar.bin
cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL --
clearFeatures yes
#cat| java           -jar ixa-pipe-nerc-1.3.6.jar tag -m $nermodel --
clearFeatures yes
◇
```

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* 24a ⟩ ≡

```
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=2babeb40a81b3720274a0521ccc2a27c5eff28c9
⟨ install from github 7a ⟩
cd $modulesdir/svm_wsd
⟨ install svm lib 24b ⟩
⟨ download svm models 24c ⟩


◇
```

Fragment referenced in 14a.

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

⟨ *install svm lib* 24b ⟩ ≡

```
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 24a.

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

⟨ *download svm models* 24c ⟩ ≡

```
⟨ get or have (24d svm_wsd.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/svm_wsd.tgz


◇
```

Fragment referenced in 24a.

Script

"../bin/wsd" 24e≡

```
#!/bin/bash
# WSD -- wrapper for word-sense disambiguation
# 8 Jan 2014 Ruben Izquierdo
# 16 sep 2014 Paul Huygen
⟨ set variables that point to the directory-structure 5a, . . . ⟩
WSDDIR=$modulesdir/svm_wsd
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $WSDDIR/$WSDSCRIPT --naf
◇
```

4.4.6 Lexical-unit converter

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

⟨ *install the lu2synset converter* 25a ⟩ ≡
```
⟨ get or have (25b lu2synset.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/lu2synset.tgz
◇
```
Fragment referenced in 14a.

Script

"../bin/lu2synset" 25c≡
```
#!/bin/bash
ROOT=$piperoot
JAVALIBDIR=$modulesdir/lexicalunitconvertor/lib
RESOURCESDIR=$modulesdir/lexicalunitconvertor/resources
JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
java -Xmx812m -
cp  $JAVALIBDIR/$JARFILE vu.wntools.util.NafLexicalUnitToSynsetReferences \
   --wn-lmf "$RESOURCESDIR/cornetto2.1.lmf.xml" --format naf
◇
```

4.4.7 NED

The NED module is rather picky about the structure of the NAF file. In any case, it does not accept a file that has been produced by the ontotagger. Hence, in a pipeline NER 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* 25d ⟩ ≡
```
⟨ put spotlight jar in the Maven repository 26a ⟩
MODNAM=ned
DIRN=ixa-pipe-ned
GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
⟨ install from github 7a ⟩
cd $modulesdir/ixa-pipe-ned
mvn -Dmaven.compiler.target=1.7 -Dmaven.compiler.source=1.7 clean package
mv target/ixa-pipe-ned-1.1.1.jar $jarsdir/
◇
```
Fragment referenced in 14a.

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* 26a ⟩ ≡

```
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 25d.

Script

"../bin/ned" 26b≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
ROOT=$piperoot
JARDIR=$jarsdir
⟨ check/start the Spotlight server 20a ⟩
cat | java -Xmx1000m -jar $jarsdir/ixa-pipe-ned-1.1.1.jar  -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 (`vua-ontotagger-v1.0.tar.gz`).

Module

⟨ *install the onto module* 26c ⟩ ≡

```
⟨ get or have (26d vua-ontotagger-v1.0.tar.gz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/vua-ontotagger-v1.0.tar.gz
rm $pipesocket/vua-ontotagger-v1.0.tar.gz
chmod -R o+r $modulesdir/vua-ontotagger-v1.0
◇
```

Fragment referenced in 14a.

Script

"../bin/onto" 27≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
ROOT=$piperoot
ONTODIR=$modulesdir/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=`mktemp -t stap6.XXXXXX`
cat >$TMPFIL

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

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

◇
```

4.4.9 Framenet SRL

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

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

"../bin/framesrl" 28a≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
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 40b.

4.4.10 Heideltime

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

One of the elements of Heideltime (the jar `de.unihd.dbs.heideltime.standalone.jar` in `NAF-HeidelTime/heidelt` has been updated and the Github version is outdated. Therefore, get the latest version from the snapshot.

⟨ *install the heideltime module* 28b ⟩ ≡

```
MODNAM=heideltime
DIRN=NAF-HeidelTime
GITU=https://github.com/cltl/NAF-HeidelTime.git
GITC=057c93ccc857a427145b9e2ff72fd645172d34df
⟨ install from github 7a ⟩
⟨ update the heideltime jar 29b ⟩
⟨ adapt heideltime's config.props 29a ⟩
◇
```

Fragment referenced in 14a.

⟨ *adapt heideltime's config.props* 29a ⟩ ≡

```
CONFIL=$modulesdir/NAF-HeidelTime/config.props
tempfil=‘mktemp -t heideltmp.XXXXXX‘
mv $CONFIL $tempfil
TREETAGDIR=treetagger
AWKCOMMAND=’/^treeTaggerHome/ {$0="treeTagger-
Home = ’$modulesdir’/treetagger"}; {print}’
gawk "$AWKCOMMAND" $tempfil >$CONFIL
rm -rf $tempfil
◇
```

Fragment referenced in 28b, 29c.
Uses: print 40b.

⟨ *update the heideltime jar* 29b ⟩ ≡

```
standalonejar=de.unihd.dbs.heideltime.standalone.jar
replstandalonejar=201506postfix.de.unihd.dbs.heideltime.standalone.jar
cd $modulesdir/NAF-HeidelTime/heideltime-standalone
rm -f $standalonejar
scp -
i "$pipesocket/nrkey" newsreader@kyoto.let.vu.nl:nlpp_resources/$replstandalonejar ./$stan-
dalonejar
◇
```

Fragment referenced in 28b.

When the installation has been transplanted, `config.props` must be updated:

⟨ *set paths after transplantation* 29c ⟩ ≡

```
⟨ adapt heideltime's config.props 29a ⟩
◇
```

Fragment referenced in 47b.

Script

"../bin/heideltime" 29d≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, . . . ⟩
HEIDELDIR=$modulesdir/NAF-HeidelTime
TEMPDIR=‘mktemp -t -d heideltmp.XXXXXX‘
cd $HEIDELDIR
iconv -t utf-
8//IGNORE | python $HEIDELDIR/HeidelTime_NafKaf.py $HEIDELDIR/heideltime-
standalone/ $TEMPDIR
rm -rf $TEMPDIR
◇
```

4.4.11 Semantic Role labelling

Module

⟨ *install the srl module* 30a ⟩ ≡

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

Fragment referenced in 14a.

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" 30b≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
source $envbindir/progenv
ROOT=$piperoot
SRLDIR=$modulesdir/vua-srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $SRLDIR
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
◇
```

File defined by 30bcde.

Create a feature-vector.

"../bin/srl" 30c≡

```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
◇
```

File defined by 30bcde.

Run the trained model on the feature-vector.

"../bin/srl" 30d≡

```
timbl -mO:I1,2,3,4 -i 25Feb2015_e-mags_mags_press_newspapers.wgt -
t $FEATUREVECTOR -o $TIMBLOUTPUTFILE >/dev/null 2>/dev/null
◇
```

File defined by 30bcde.

Insert the SRL values into the NAF file.

"../bin/srl" 30e≡

```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
◇
```

File defined by 30bcde.

Clean up.

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* 31a ⟩ ≡
```
⟨ get or have (31b 20150706vua-srl-dutch-additional-roles.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/20150706vua-srl-dutch-additional-roles.tgz
◇
```
Fragment referenced in 14a.

⟨ *clean up* 31c ⟩ ≡
```
rm -rf $pipesocket/20150706vua-srl-dutch-additional-roles.tgz
◇
```
Fragment defined by 9c, 10b, 15f, 31c, 37c.
Fragment referenced in 36d.

Script

"../bin/postsrl" 31d≡
```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
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* 31e ⟩ ≡
```
⟨ get or have (31f 20150702-vua-eventcoreference_v2.tgz ) 8a ⟩
cd $modulesdir
tar -xzf $pipesocket/20150702-vua-eventcoreference_v2.tgz
cd vua-eventcoreference_v2
cp lib/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar $jarsdir
◇
```
Fragment referenced in 14a.

Script

```
"../bin/evcoref" 32a≡
        #!/bin/bash
        ⟨ set variables that point to the directory-structure 5a, ... ⟩
        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

Module

```
⟨ install the dbpedia-ner module 32b ⟩ ≡
        MODNAM=dbpedia_ner
        DIRN=dbpedia_ner
        GITU=https://github.com/PaulHuygen/dbpedia_ner.git
        GITC=ab1dcbd860f0ff29bc979f646dc382122a101fc2
        ⟨ install from github 7a ⟩
        ◇
```

Fragment referenced in 14a.

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" 32c≡
        #!/bin/bash
        ⟨ set variables that point to the directory-structure 5a, ... ⟩
        ⟨ check/start the Spotlight server 20a ⟩
        MODDIR=$modulesdir/dbpedia_ner
        cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
        url http://localhost:2060/rest/candidates
        ◇
```

4.5 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* 33a ⟩ ≡
⟨ *get or have* (33b `vua-postprocess-nl.zip`) 8a ⟩
```
cd $modulesdir
unzip -q $pipesocket/vua-postprocess-nl.zip
◇
```
Fragment referenced in 14a.

Script

"../bin/nomevent" 33c≡
```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
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
◇
```

5 Utilities

5.1 Test script

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

"../bin/test" 33d≡
```
#!/bin/bash
ROOT=/mnt/sdb1/pipelines/testnlpp/nlpp
TESTDIR=$ROOT/test
BIND=$ROOT/bin
mkdir -p $TESTDIR
cd $TESTDIR
cat $ROOT/nuweb/testin.naf   | $BIND/tok                    > $TESTDIR/test.tok.naf
cat test.tok.naf             | $BIND/mor                    > $TESTDIR/test.mor.naf
cat test.mor.naf             | $BIND/nerc_conll02 > $TESTDIR/test.nerc.naf
cat $TESTDIR/test.nerc.naf   | $BIND/wsd                    > $TESTDIR/test.wsd.naf
cat $TESTDIR/test.wsd.naf    | $BIND/ned                    > $TESTDIR/test.ned.naf
cat $TESTDIR/test.ned.naf    | $BIND/onto                   > $TESTDIR/test.onto.naf
cat $TESTDIR/test.onto.naf   | $BIND/heideltime             > $TESTDIR/test.times.naf
cat $TESTDIR/test.times.naf  | $BIND/srl                    > $TESTDIR/test.srl.naf
cat $TESTDIR/test.srl.naf    | $BIND/evcoref       > $TESTDIR/test.ecrf.naf
cat $TESTDIR/test.ecrf.naf   | $BIND/framesrl      > $TESTDIR/test.fsrl.naf
cat $TESTDIR/test.fsrl.naf   | $BIND/dbpner        > $TESTDIR/test.dbpner.naf
cat $TESTDIR/test.dbpner.naf | $BIND/nomevent      > $TESTDIR/test.nomev.naf
cat $TESTDIR/test.nomev.naf  | $BIND/postsrl       > $TESTDIR/test.psrl.naf
◇
```
Uses: `nuweb` 42d.

5.2 Logging

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

⟨ *variables of install-modules* 34a ⟩ ≡
```
LOGLEVEL=1
```
◇

Fragment referenced in 13.

⟨ *logmess* 34b ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
```
◇

Fragment referenced in 6c, 7a, 12b, 15b, 18a, 34c.

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* 34c ⟩ ≡
```
SUCCES=0
cd $modulesdir
⟨ move module (34d $DIR ) 6a ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (34e Installed $DIR ) 34b ⟩
  ⟨ remove old module (34f $DIR ) 6b ⟩
else
  ⟨ re-instate old module (34g $DIR ) 6c ⟩
fi
```
◇

Fragment never referenced.

A How to read and translate this document

This document is an example of *literate programming* [1]. It contains the code of all sorts of scripts and programs, combined with explaining texts. In this document the literate programming tool `nuweb` is used, that is currently available from Sourceforge (URL:`nuweb.sourceforge.net`). The advantages of Nuweb are, that it can be used for every programming language and scripting language, that it can contain multiple program sources and that it is very simple.

A.1 Read this document

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

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

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

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

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

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

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

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

A.2 Process the document

The raw document is named `a_nlpp.w`. Figure 2 shows pathways to translate it into print-

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

able/viewable documents and to extract the program sources. Table 3 lists the tools that are needed for a translation. Most of the tools (except Nuweb) are available on a well-equipped Linux system.

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

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

⟨ *parameters in Makefile* 36a ⟩ ≡
```
NUWEB=../env/bin/nuweb
```
◇

Fragment defined by 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Uses: nuweb 42d.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

"Makefile" 36b≡
⟨ *default target* 36c ⟩

⟨ *parameters in Makefile* 36a, ... ⟩

⟨ *impliciete make regels* 39c, ... ⟩
⟨ *expliciete make regels* 37b, ... ⟩
⟨ *make targets* 36d, ... ⟩
◇

The default target of make is `all`.

⟨ *default target* 36c ⟩ ≡
```
all : ⟨ all targets 36e ⟩
.PHONY : all
```
◇

Fragment referenced in 36b.
Defines: all Never used, PHONY 40a.

⟨ *make targets* 36d ⟩ ≡
```
clean:
```
⟨ *clean up* 9c, ... ⟩
◇

Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.

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

⟨ *all targets* 36e ⟩ ≡
```
nlpp.pdf
```
◇

Fragment referenced in 36c.
Uses: pdf 40b.

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

⟨ *parameters in Makefile* 37a ⟩ ≡

```
.SUFFIXES: .pdf .w .tex .html .aux .log .php
```

◇

Fragment defined by 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 40b.

A.4 Get Nuweb

An annoying problem is, that this program uses nuweb, a utility that is seldom installed on a computer. Therefore, we are going to install that first if it is not present. Unfortunately, nuweb is hosted on sourceforge and it is difficult to achieve automatic downloading from that repository. Therefore I copied one of the versions on a location from where it can be downloaded with a script.

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

⟨ *expliciete make regels* 37b ⟩ ≡

```
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 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.
Uses: `nuweb` 42d.

⟨ *clean up* 37c ⟩ ≡

```
rm -rf ../nuweb-1.58
```

◇

Fragment defined by 9c, 10b, 15f, 31c, 37c.
Fragment referenced in 36d.
Uses: `nuweb` 42d.

⟨ *expliciete make regels* 37d ⟩ ≡

```
../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 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.
Uses: `nuweb` 42d.

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* 38a ⟩ ≡

```
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 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.
Uses: `print` 40b.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 38b ⟩ ≡

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

◇

Fragment defined by 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.

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* 39a ⟩ ≡

```
FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Defines: `FIGFILES` 39b.

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* 39b ⟩ ≡

```
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 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 41a, `PDF_FIG_NAMES` 41a, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 39a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 39c ⟩ ≡

```
%.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 39c, 44a.
Fragment referenced in 36b.
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* 40a ⟩ ≡

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

.PHONY : bibfile
◇
```

Fragment defined by 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.
Uses: PHONY 36c.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 40b ⟩ ≡

```
pdf : nlpp.pdf

print : nlpp.pdf
        lpr nlpp.pdf

view : nlpp.pdf
        evince nlpp.pdf

◇
```

Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.
Defines: `pdf` 36e, 37a, 41a, `print` 10d, 28a, 29a, 38a, `view` Never used.

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

The w2pdf script The three processors nuweb, LaTeX and bibTeX are intertwined. LaTeX and bibTeX create parameters or change the value of parameters, and write them in an auxiliary file. The other processors may need those values to produce the correct output. The LaTeX processor may even need the parameters in a second run. Therefore, consider the creation of the (PDF) document finished when none of the processors causes the auxiliary file to change. This is performed by a shell script `w2pdf`.

⟨ *make targets* 41a ⟩ ≡
```
nlpp.pdf : nlpp.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

```
◇
Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.
Uses: `pdf` 40b, `PDFT_NAMES` 39b, `PDF_FIG_NAMES` 39b.

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* 41b ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 4abc, 8c, 9ef, 11f, 41b.
Fragment referenced in 46b.
Uses: `nuweb` 42d.

⟨ *parameters in Makefile* 41c ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Uses: `nuweb` 42d.

⟨ *expliciete make regels* 41d ⟩ ≡
```
$(W2PDF) : nlpp.w $(NUWEB)
        $(NUWEB) nlpp.w
◇
```
Fragment defined by 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.

`"../nuweb/bin/w2pdf"` 41e≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20150708 at 0957h: Generated by nuweb from a_nlpp.w
NUWEB=../env/bin/nuweb
LATEXCOMPILER=pdflatex
⟨ filenames in nuweb compile script 42b ⟩
⟨ compile nuweb 42a ⟩

◇
```
Uses: `nuweb` 42d.

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

⟨ *compile nuweb* 42a ⟩ ≡
```
NUWEB=/mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/nuweb
⟨ run the processors until the aux file remains unchanged 43a ⟩
⟨ remove the copy of the aux file 42c ⟩
◇
```
Fragment referenced in 41e.
Uses: nuweb 42d.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. `.w`) from the filename and create the names of the LaTeX file (ends with `.tex`), the auxiliary file (ends with `.aux`) and the copy of the auxiliary file (add `old.` as a prefix to the auxiliary filename).

⟨ *filenames in nuweb compile script* 42b ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 41e.
Defines: auxfil 43a, 45ab, indexfil 43a, 45a, nufil 42d, 45ac, oldaux 42c, 43a, 45ab, oldindexfil 43a, 45a, texfil 42d, 45ac, trunk 42d, 45acd.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 42c ⟩ ≡
```
rm $oldaux
◇
```
Fragment referenced in 42a, 44e.
Uses: oldaux 42b, 45a.

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

⟨ *run the three processors* 42d ⟩ ≡
```
$NUWEB $nufil
$LATEXCOMPILER $texfil
makeindex $trunk
bibtex $trunk
◇
```
Fragment referenced in 43a.
Defines: bibtex 45cd, makeindex 45cd, nuweb 5a, 33d, 36a, 37bcd, 41bce, 42a, 43b, 44d.
Uses: nufil 42b, 45a, texfil 42b, 45a, trunk 42b, 45a.

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* 43a ⟩ ≡

```
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 42d ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
◇
```

Fragment referenced in 42a.
Uses: `auxfil` 42b, 45a, `indexfil` 42b, `oldaux` 42b, 45a, `oldindexfil` 42b.

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* 43b ⟩ ≡

```
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 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Uses: `nuweb` 42d.

Make the directory:

⟨ *expliciete make regels* 43c ⟩ ≡

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

◇
```

Fragment defined by 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.

The rule to copy files in it:

⟨ *impliciete make regels* 44a ⟩ ≡
```
$(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
```
◇
Fragment defined by 39c, 44a.
Fragment referenced in 36b.

Do the work:

⟨ *expliciete make regels* 44b ⟩ ≡
```
$(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
```
◇
Fragment defined by 37bd, 38ab, 40a, 41d, 43c, 44b.
Fragment referenced in 36b.

Invoke:

⟨ *make targets* 44c ⟩ ≡
```
htm : $(htmldir) $(htmltarget)
```
◇
Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.

Create a script that performs the translation.

"w2html" 44d≡
```
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
#  [filename]: Name of the nuweb source file.
# 20150708 at 0957h: Generated by nuweb from a_nlpp.w
echo "translate " $1 >w2html.log
NUWEB=/mnt/sdb1/pipelines/testnlpp/nlpp/env/bin/nuweb
```
⟨ *filenames in w2html* 45a ⟩

⟨ *perform the task of w2html* 44e ⟩

◇
Uses: nuweb 42d.

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* 44e ⟩ ≡
⟨ *run the html processors until the aux file remains unchanged* 45b ⟩
⟨ *remove the copy of the aux file* 42c ⟩
◇
Fragment referenced in 44d.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. `.w`) from the filename and create the names of the LaTeX file (ends with `.tex`), the auxiliary file (ends with `.aux`) and the copy of the auxiliary file (add `old.` as a prefix to the auxiliary filename).

⟨ *filenames in w2html* 45a ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 44d.
Defines: `auxfil` 42b, 43a, 45b, `nufil` 42bd, 45c, `oldaux` 42bc, 43a, 45b, `texfil` 42bd, 45c, `trunk` 42bd, 45cd.
Uses: `indexfil` 42b, `oldindexfil` 42b.

⟨ *run the html processors until the aux file remains unchanged* 45b ⟩ ≡
```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 45c ⟩
done
⟨ run tex4ht 45d ⟩

◇
```
Fragment referenced in 44e.
Uses: `auxfil` 42b, 45a, `oldaux` 42b, 45a.

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

⟨ *run the html processors* 45c ⟩ ≡
```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 45b.
Uses: `bibtex` 42d, `makeindex` 42d, `nufil` 42b, 45a, `texfil` 42b, 45a, `trunk` 42b, 45a.

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* 45d ⟩ ≡
```
tex '\def\filename{{nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 45b.
Uses: `bibtex` 42d, `makeindex` 42d, `trunk` 42b, 45a.

A.7 Create the program sources

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

⟨ *parameters in Makefile* 46a ⟩ ≡
```
MKDIR = mkdir -p
```
◇
Fragment defined by 36a, 37a, 39ab, 41c, 43b, 46a.
Fragment referenced in 36b.
Defines: MKDIR 46b.

⟨ *make targets* 46b ⟩ ≡
```
DIRS = ⟨ directories to create 4a, ... ⟩

$(DIRS) :
        $(MKDIR) $@
```
◇
Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.
Defines: DIRS 46d.
Uses: MKDIR 46a.

⟨ *make scripts executable* 46c ⟩ ≡
```
chmod -R 775  ../bin/*
chmod -R 775  ../env/bin/*
```
◇
Fragment defined by 14b, 46c.
Fragment referenced in 46d.

⟨ *make targets* 46d ⟩ ≡
```
sources : nlpp.w $(DIRS) $(NUWEB)
        $(NUWEB) nlpp.w
        ⟨ make scripts executable 14b, ... ⟩
```
◇
Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.
Uses: DIRS 46b.

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* 47a ⟩ ≡

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

```

◇

Fragment defined by 36d, 40b, 41a, 44c, 46bd, 47a.
Fragment referenced in 36b.

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

"../env/bin/transplant" 47b≡

```
#!/bin/bash
⟨ set variables that point to the directory-structure 5a, ... ⟩
⟨ set paths after transplantation 29c ⟩
⟨ re-install modules after the transplantation 18d ⟩

```

◇

B References

B.1 Literature

References

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

C Indexes

C.1 Filenames

C.2 Macro's

⟨ activate the python environment 11e, 12a ⟩ Referenced in 10c.
⟨ adapt heideltime's config.props 29a ⟩ Referenced in 28b, 29c.
⟨ all targets 36e ⟩ Referenced in 36c.
⟨ check this first 7f, 14c ⟩ Referenced in 13.
⟨ check whether mercurial is present 15a ⟩ Referenced in 14c.
⟨ check/install the correct version of python 10d ⟩ Referenced in 10c.
⟨ check/start the Spotlight server 20a ⟩ Referenced in 26b, 32c.
⟨ clean up 9c, 10b, 15f, 31c, 37c ⟩ Referenced in 36d.
⟨ compile nuweb 42a ⟩ Referenced in 41e.
⟨ compile the nerc jar 22b ⟩ Referenced in 22a.
⟨ create a virtual environment for Python 11c ⟩ Referenced in 10c.
⟨ create progenv script 8b ⟩ Referenced in 13.
⟨ default target 36c ⟩ Referenced in 36b.
⟨ directories to create 4abc, 8c, 9ef, 11f, 41b ⟩ Referenced in 46b.
⟨ download svm models 24c ⟩ Referenced in 24a.
⟨ expliciete make regels 37bd, 38ab, 40a, 41d, 43c, 44b ⟩ Referenced in 36b.
⟨ filenames in nuweb compile script 42b ⟩ Referenced in 41e.
⟨ filenames in w2html 45a ⟩ Referenced in 44d.
⟨ get or have 8a ⟩ Referenced in 9a, 11a, 15b, 19a, 23a, 24c, 25a, 26c, 31ae, 33a.
⟨ get the nerc models 23a ⟩ Referenced in 22a.
⟨ impliciete make regels 39c, 44a ⟩ Referenced in 36b.
⟨ install ActivePython 11a ⟩ Referenced in 10d.
⟨ install Alpino 15b ⟩ Referenced in 13.
⟨ install coreference-base 21c ⟩ Referenced in 14a.
⟨ install from github 7a ⟩ Referenced in 21ac, 24a, 25d, 28b, 30a, 32b.
⟨ install from tarball 34c ⟩ Not referenced.
⟨ install kafnafparserpy 12b ⟩ Referenced in 10c.
⟨ install maven 9g, 10a ⟩ Referenced in 13.
⟨ install python packages 12g ⟩ Referenced in 10c.
⟨ install svm lib 24b ⟩ Referenced in 24a.
⟨ install the dbpedia-ner module 32b ⟩ Referenced in 14a.
⟨ install the event-coreference module 31e ⟩ Referenced in 14a.
⟨ install the heideltime module 28b ⟩ Referenced in 14a.
⟨ install the lu2synset converter 25a ⟩ Referenced in 14a.
⟨ install the morphosyntactic parser 21a ⟩ Referenced in 14a.
⟨ install the NERC module 22a ⟩ Referenced in 14a.
⟨ install the nomevent module 33a ⟩ Referenced in 14a.
⟨ install the onto module 26c ⟩ Referenced in 14a.
⟨ install the post-SRL module 31a ⟩ Referenced in 14a.
⟨ install the Spotlight server 19ac ⟩ Referenced in 13.
⟨ install the srl module 30a ⟩ Referenced in 14a.
⟨ install the ticcutils utility 17c ⟩ Referenced in 13, 18d.
⟨ install the timbl utility 17d ⟩ Referenced in 13, 18d.
⟨ install the tokenizer 20b ⟩ Referenced in 14a.
⟨ install the treetagger utility 16abcde, 17ab ⟩ Referenced in 13.
⟨ install the WSD module 24a ⟩ Referenced in 14a.
⟨ install the NED module 25d ⟩ Referenced in 14a.
⟨ logmess 34b ⟩ Referenced in 6c, 7a, 12b, 15b, 18a, 34c.
⟨ make scripts executable 14b, 46c ⟩ Referenced in 46d.
⟨ make targets 36d, 40b, 41a, 44c, 46bd, 47a ⟩ Referenced in 36b.
⟨ move module 6a ⟩ Referenced in 7a, 12b, 34c.
⟨ parameters in Makefile 36a, 37a, 39ab, 41c, 43b, 46a ⟩ Referenced in 36b.
⟨ perform the task of w2html 44e ⟩ Referenced in 44d.
⟨ put spotlight jar in the Maven repository 26a ⟩ Referenced in 25d.
⟨ re-install modules after the transplantation 18d ⟩ Referenced in 47b.
⟨ re-instate old module 6c ⟩ Referenced in 7a, 12b, 34c.
⟨ remove old module 6b ⟩ Referenced in 7a, 12b, 34c.

⟨ remove the copy of the aux file 42c ⟩ Referenced in 42a, 44e.
⟨ run tex4ht 45d ⟩ Referenced in 45b.
⟨ run the html processors 45c ⟩ Referenced in 45b.
⟨ run the html processors until the aux file remains unchanged 45b ⟩ Referenced in 44e.
⟨ run the processors until the aux file remains unchanged 43a ⟩ Referenced in 42a.
⟨ run the three processors 42d ⟩ Referenced in 43a.
⟨ set alpinohome 15e ⟩ Referenced in 21b.
⟨ set paths after transplantation 29c ⟩ Referenced in 47b.
⟨ set up java 9ad ⟩ Referenced in 13.
⟨ set up python 10c ⟩ Referenced in 13.
⟨ set variables that point to the directory-structure 5abc ⟩ Referenced in 13, 20c, 21bd, 23cd, 24e, 26b, 27, 28a, 29d, 30b, 31d, 32ac, 33c, 47b.
⟨ start the Spotlight server 19d ⟩ Referenced in 20a.
⟨ test whether virtualenv is present on the host 11d ⟩ Referenced in 11c.
⟨ unpack ticcutils or timbl 18a ⟩ Referenced in 17cd.
⟨ update the heideltime jar 29b ⟩ Referenced in 28b.
⟨ variables of install-modules 34a ⟩ Referenced in 13.

C.3 Variables

`activate`: 11e.
`all`: 36c.
`ALPINO_HOME`: 15e.
`auxfil`: 42b, 43a, 45a, 45b.
`bibtex`: 42d, 45cd.
`DIRS`: 46b, 46d.
`fig2dev`: 39c.
`FIGFILENAMES`: 39b.
`FIGFILES`: 39a, 39b.
`hg`: 15a, 21c.
`indexfil`: 42b, 43a, 45a.
`lxml`: 12g.
`makeindex`: 42d, 45cd.
`MKDIR`: 46a, 46b.
`nufil`: 42b, 42d, 45a, 45c.
`nuweb`: 5a, 33d, 36a, 37bcd, 41bce, 42a, 42d, 43b, 44d.
`oldaux`: 42b, 42c, 43a, 45a, 45b.
`oldindexfil`: 42b, 43a, 45a.
`PATH`: 5b, 9d, 10a.
`pdf`: 36e, 37a, 40b, 41a.
`PDFT_NAMES`: 39b, 41a.
`PDF_FIG_NAMES`: 39b, 41a.
`PHONY`: 36c, 40a.
`print`: 10d, 28a, 29a, 38a, 40b.
`PST_NAMES`: 39b.
`PS_FIG_NAMES`: 39b.
`pythonok`: 10d.
`PYTHONPATH`: 12a.
`pyyaml`: 12g.
`SUFFIXES`: 37a.
`texfil`: 42b, 42d, 45a, 45c.
`trunk`: 42b, 42d, 45a, 45cd.
`view`: 40b.
`virtualenv`: 11ac, 11d.