WikiTailor

Technical Manual

Version 0.1

Cristina España-Bonet, Alberto Barrón-Cedeño and Josu Boldoba

WORK IN PROGRESS

March 9, 2016

Abstract

WikiTailor is a toolkit designed to extract and analise corpora from Wikipedia in any language and domain[1]. It is intended to be a useful tool for researchers and practitioners interested in exploiting (domain-specific) fractions on Wikipedia both in mono- and multilingual settings.

The toolkit consists of a suite of Java programs which can be used through a stand-alone executable jar file. This document describes the prerequisites and the usage of the toolkit, the main characteristics of the systems involved in the extraction of the corpora, and some methods to evaluate the extractions.

[1]This work has been partially funded by the TACARDI project (TIN2012-38523-C02) of the Spanish Ministerio de Economía y Competitividad (MEC).

Contents

Acronyms

JWPL Java Wikipedia Library. 3, 4

RDBMS Relational Database Management System. 4

Glossary

DATE Date of a Wikipedia dump in the `YEARMONTHDAY` format (e.g., 20150225).. 3, 4

DISAMBIGUATION_CATEGORY_NAME Name of the category that contains the disambiguation categories.. 4

LAN Two-characters ISO 639 language code[2] es un link a iso2, hay que usar iso1 si existe, el 2 sino. 3, 4

LANGUAGE String matching one in `languages.txt` in this release, corresponding to the full name of the language.. 4

MAIN_CATEGORY_NAME Name of the main (top) category of the Wikipedia category hierarchy.. 4

YEAR Year of a Wikipedia dump.. 4

Acronyms

JWPL Java Wikipedia Library. 3, 4

RDBMS Relational Database Management System. 4

[2] `http://www.loc.gov/standards/iso639-2/php/code_list.php` ISO 639-1 is used when available. no entiendo la precision

1 Introduction

2 Prerequisites and Installation for the Executable

This software is distributed as a stand-alone executable jar file on the following URL:

```
http://cristinae.github.io/WikiTailor/dwnld/wikiTailor-v1.0.0.with-dependencies.tar.gz
```

It requires Java version 7 or higher. The source code is available as well on Github:

```
https://github.com/cristinae/WikiTailor
```

The compilation system is built with Apache's ant[3] (cf. Section 2.3).

A local copy of the Wikipedia in the desired language editions is required. Sections 2.1 and 2.2 describe how to download the necessary dumps and how to pre-process the data to store in a in-house database, as required by WikiTailor.

2.1 Dumps Acquisition

Wikimedia makes available regularly-generated dumps of its contents in multiple languages. Such dumps ara available for download at the following URL:

```
http://dumps.wikimedia.org/[LAN]wiki/[DATE]
```

WikiTailor requires the following dump files:

```
[LAN]wiki-[DATE]-pages-articles.xml.bz2
[LAN]wiki-[DATE]-pagelinks.sql.gz
[LAN]wiki-[DATE]-categorylinks.sql.gz
[LAN]wiki-[DATE]-langlinks.sql.gz
[LAN]wiki-[DATE]-page.sql.gz
[LAN]wiki-[DATE]-redirect.sql.gz
```

The former three files contain the tables required by the Java Wikipedia Library (JWPL). Before feeding them into the MySQL RDBMS, a pre-processing step is necessary. This procedure is described in the next sub-section. The latter three files contain the necessary information to relate articles across multiple language editions and to handle re-directions. No pre-processing is necessary in this case and the SQL tables can be fed directly into the database.

2.2 The JWPL DataMachine and the Database

WikiTailor relies on JWPL to get off-line access to the Wikipedia contents [2]. Therefore, pre-processing the data with JWPL's DataMachine module is necessary to parse the Wikipedia dumps and make them accessable to WikiTailor.
The full dump preprocessing consists of the following steps:

Step 1. Download JWPL DataMachine and the `tables.sql` file from

[3] `http://ant.apache.org/`

https://dkpro.github.io/dkpro-jwpl/

Step 2. Database setup. Create a MySQL database using UTF-8 encoding and feed the `tables.sql` files to generate the necessary tables:

```
mysql> CREATE DATABASE IF NOT EXISTS [DBname]  CHARACTER SET  utf8 \
COLLATE utf8_general_ci

$ mysql -u [USER] -p [DBname] < tables.sql
```

where `DBname` should have the form `[DBrootname]_wiki_[LAN]_[YEAR]`, as expected by WikiTailor.

Step 3. JWPL pre-processing. Run the transformation:

```
$ java -jar -Xmx4g datamachine.jar [LANGUAGE] \
[CATEGORY:MAIN_CATEGORY_NAME] \
[CATEGORY:DISAMBIGUATION_CATEGORY_NAME] [SOURCE_DIRECTORY]
```

where:
`LANGUAGE` is a string matching one in `languages.txt` in this release,
`MAIN_CATEGORY_NAME` is the name of the main (top) category of the Wikipedia category hierarchy,
`DISAMBIGUATION_CATEGORY_NAME` is the name of the category that contains the disambiguation categories and,
`SOURCE_DIRECTORY` is the path to the directory containing the data dumps.

The values for these fields in the languages currently implemented in WikiTailor are included in Appendix A

This step should create the following new data files in an *./output* subfolder:

```
category_inlinks.txt
category_outlinks.txt
category_pages.txt
Category.txt
MetaData.txt
page_categories.txt
page_inlinks.txt
PageMapLine.txt
page_outlinks.txt
page_redirects.txt
Page.txt
```

Examples:

```
java -jar -Xmx4g \
de.tudarmstadt.ukp.wikipedia.datamachine-1.0-jar-with-dependencies.jar\
english Category:Contents Category:All_disambiguation_pages ./en/
```

```
java -jar -Xmx2g \
de.tudarmstadt.ukp.wikipedia.datamachine-1.0-jar-with-dependencies.jar\
catalan Categoria:Principal Categoria:Pàgines_de_desambiguació ./ca/
```

```
java -jar -Xmx2g \
de.tudarmstadt.ukp.wikipedia.datamachine-1.0-jar-with-dependencies.jar\
arabic توضيح _ويكيبيديا تصنيف: صفحات _محتويات تصنيف: ./ar/
```

Step 4. Finally, import the generated data files into de database.

```
$ mysqlimport -u[USER] -h[HOST] -p --local --default-character-set=utf8 \
[DBname] *.txt
```

A MySQL database can host the tables related to the interlanguage links, the langlinks. Just modify the names of the tables in the downloaded files to fit the WikiTailor nomenclature and upload them into the database.

Step 1. Create the database:

```
mysql> CREATE DATABASE IF NOT EXISTS [DBrootname_pairs] CHARACTER SET utf8 \
COLLATE utf8_general\ci
```

Step 2. Within the sql file, change the name of the table ‘`langlinks`‘ into ‘`wiki[LAN]_[YEAR]_langlinks`‘. Do the equivalent modification for ‘`page`‘ and ‘`redirect`‘.

Step 3. Upload the tables:

```
$ mysql -u[USER] -p[DBrootname_pairs] < [LAN]wiki-[DATE]-langlinks.modified.sql
$ mysql -u[USER] -p[DBrootname_pairs] < [LAN]wiki-[DATE]-page.modified.sql
$ mysql -u[USER] -p[DBrootname_pairs] < [LAN]wiki-[DATE]-redirect.modified.sql
```

2.3 WikiTailor Source Code Installation

For those interested in going beyond just using the jar program, we now present the instructions to setup the WikiTailor Java project. The described procedure is valid for Linux boxes. We have evidence of successful installations in OSx as well. The pre-requisites for the setup include Git, Apache's Ant and Java 7 or higher. In Ubuntu-like distributions, the commands to install them are:

```
$ sudo apt-get install git
$ sudo apt-get install ant
$ sudo apt-get install openjdk-7-jdk
```

2.3.1 Building WikiTailor

The up-to-date version of the WikiTailor is available on Github. To clone it, run the following command:

```
$ git clone https://github.com/cristinae/WikiTailor
```

cd into the WikiTailor folder and run the following ant commands:

```
$ ant deploy-src
$ ant testNoDB
```

The first command compiles the source code, whereas the second one runs the basic tests to check everything is up and running. If no errors appear, you are ready to use WikiTailor. Clean the installation, cd into the distribution folder, and see the following section:

```
$ ant clean
$ cd WT-v$version
```

2.3.2 External Dependencies

WikiTailor requires the following Java libraries, all of them under open source licenses:

- apfloat-1.8.1
- apache-commons-cli-1.2
- apache-commons-collections-4.4.0
- icu4j-4.8.1.1
- apache-commons-io-2.4
- apache-commons-lang3-3.2.1
- apache-log4j-2.0
- apache-lucene
- jama-1.0.3
- junit-4.11
- jwpl-0.9.2
- apache-commons-math3-3.4.1
- opencsv-2.3
- apache-opennlp-1.5.3
- snowball-stemmer
- gnu-trove-3.0.3

Currently these libraries are included in the git repository. As a result, the compilation of the project can be launched right away.

3 Description and Usage

In this section we describe how to use WikiTailor. We start with the initial setup in Section 3.1. Section 3.2 explains how to use the two implemented models to extract in-domain corpora from Wikipedia.

3.1 Initial Setup

Once the desired Wikipedia edition has been stored in a MySQL database (Section 2.2), WikiTailor requires some configuration information to properly handle the data in the required language edition. Figure 1 displays the location of the configuration files.

wikiTailor.ini is the main configuration file and is located at the root of the package. Besides the values for these parameters, the nomenclature for the output files is also specified. The names

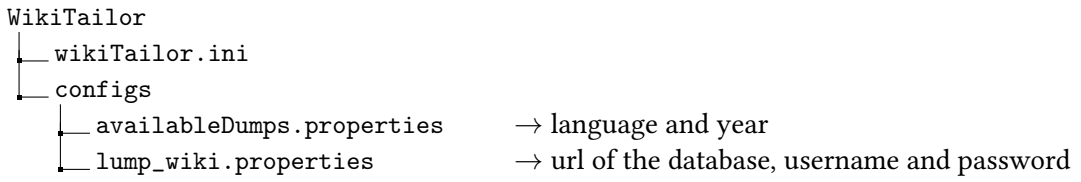

Figure 1: Folder structure of WikiTailor.

for the files include placeholders that WikiTailor fills with information about the model being run; these placeholders cannot be modified.

```
################################################
###   WikiTailor configuration file
################################################

### Graph-based extraction
# Model parameters: category graph
percentage=0.5
minDepth=0
maxDepth=50
# Model parameters: domain keywords
topPercentage=10
topKeywords=100
minNumArticles=10
# Nomenclature (Do not change the place holders)
categoryFileName = %s.%d.%d.category
articlesFileName = %s.%d.%d.articles
dictFileName = %s.%d.dict
statsFileName = %s.%d.stats

### IR-based extraction
# Model parameters
minPercentage=10
# WT domain keywords
topPercentage4L=10
topKeywords4L=100
minNumArticles4L=10
# Nomenclature (Do not change the place holders)
eArticlesFileName = %s.%s.extracted.articles
```

3.2 Domain Articles Extraction

The two models available in WikiTailor can be executed from beginning to end using an *Xecutor* class, implemented within their corresponding packages[4]. In both cases, each step of the process can be excecuted individually. This can be made either by limiting the required steps to the Xecutor with the command line arguments *-start* and *-end*, or by using the specific class in charge of the desired task. Following, we describe how the full pipeline for each system works.

3.2.1 Graph-based System

The Xecutor for the graph-based system is the main class of WikiTailor. It can be accessed by:

[4]We recommend to extract the articles with the graph-based model since it produces better results [1].

```
user@machine:~/WT-v1.0.0/java -jar wikiTailor.v1.0.0.light.jar -h

usage: java -jar wikiTailor.v1.0.0.light.jar [-c <arg> | -n <arg>] [-d <arg>] [-e <arg>]
       [-h] -i <FILE> -l <arg> [-m <arg>]  [-o <arg>] [-s <arg>] [-t <arg>] -y <arg>

where the arguments are:
 -c,--category <arg>      Name of the category (with '_' instead of ' '; you can use -n
                          instead)
 -d,--depth <arg>         Depth obtained in a previous execution
                          (default: 0)
 -e,--end <arg>           Last step for the process
                          (default: 7)
 -h,--help                This help
 -i,--ini <FILE>          Global config file for WikiTailor
 -l,--language <arg>      Language of interest (e.g., en, es, ca...)
 -m,--model <arg>         Percentage of in-domain categories
                          (default: 0.5)
 -n,--numcategory <arg>   Numerical identifier of the category (you can use -c instead)
 -o,--outpath <arg>       Save the output into this directory
                          (default: current)
 -s,--start <arg>         Initial step for the process
                          (default: 1)
 -t,--top <arg>           Number of vocabulary terms within the 10%
                          (default: 100, all: -1)
 -y,--year <arg>          Wikipedia year edition (e.g., 2015, 2016...)

Ex: java -jar wikiTailor.v1.0.0.light.jar -l en -y 2015 -i wikiTailor.ini -c Science
```

There are two mandatory arguments that define the Wikipedia edition to use, *-language* and *-year*. The configuration file that specifies the default parameters for the model is loaded with the *-ini* option. The most relevant parameters for the model can be modified via the command line as seen above. The domain for which you want to extract the articles must be detailed using the *-category* or *-numcategory* options. In the first case, the domain is characterised by the title of a category existing in Wikipedia; in the second case, the associated ID is used.

By default, the Xecutor runs seven modules:

Step 1. **DomainKeywords.** Extraction of the vocabulary as a set of lemmas associated to the domain. The *-top* parameter controls the size of the vocabulary (running it with $t = 100$ should be enough). The output of this step is a file with extension *.dict* with the list of lemmas; it is used in step 3.

Step 2. **CategoryExtractor.** Identification of the top-50 levels (if available) of subcategories from the desired root category. The output is a file listing all those categories with extension *.50.categories* and format:
`level catID catTitle`
This information is used in steps 3 and 4.

Step 3. **CategoryNameStats.** Estimation of the percentage of in-domain category titles per level of the tree (given the vocabulary generated in step 1 and the categories' tree generated in step 2). The levels together with the associated percentage of positive articles (i.e. belonging to the domain), are stored in a file with extension *.stats*.

Step 4. **CategoryDepth.** Search for the depth up to which articles belong to the domain in the category tree. This is defined by the parameter *-model*, used to select the percentage of positive articles accepted for considering a level as in-domain. The output of this step is an integer with the selected depth that can be given to the following steps using the *-depth* parameter.

Step 5. **createCategoriesFile.** Extraction of all the subcategories from the desired category up to the depth obtained in step 4. It uses the file of step 2 and selects the adequate ones. As in step 2, the result is a list of the categories with extension *.$depth.categories* and format: `level catID catTitle`

Step 6. **ArticleSelector.** Extraction of a list with the IDs of the articles associated to a tree of categories up to the selected depth. The categories' tree generated in step 5 is needed.

Step 7. **ArticleTextExtractor.** Extracts the in-domain articles' contents into plain text files. The list or articles produced in step 6 is needed.

3.2.2 IR-based System

As for the graph-based system, there is an *Xecutor* class that controls the execution of the IR-based system. It can be accessed by:

```
user@machine:~/WT-v1.0.0/java -cp wikiTailor.v1.0.0.light.jar cat.lump.ir.lucene.Xecutor

usage: java -cp wikiTailor.v1.0.0.light.jar cat.lump.ir.lucene.Xecutor [-c <arg> | -n
       <arg>] [-d <arg>] [-e <arg>] [-f <FILE>] [-h] -i <FILE> -l <arg>  [-o <arg>] [-s
       <arg>] [-t <arg>] [-x <arg>] -y <arg>

where the arguments are:
 -c,--category <arg>      Name of the category (with '_' instead of ' '; you can use -n
                          instead)
 -d,--inputDir <arg>      Directory to store/read the raw Wikipedia articles
 -e,--end <arg>           Last step for the process
                          (default: 5)
 -f,--dictFile <FILE>     Vocabulary file for the category (can be estimated as a first
                          step)
 -h,--help                This help
 -i,--ini <FILE>          Global config file for WikiTailor
 -l,--language <arg>      Language of interest (e.g., en, es, ca...)
 -n,--numcategory <arg>   Numerical identifier of the category (you can use -c instead)
 -o,--outputDir <arg>     Save the output into this directory
                          (default: current)
 -s,--start <arg>         Initial step for the process
                          (default: 1)
 -t,--top <arg>           Number of vocabulary terms within the 10%
                          (default: 100, all: -1)
 -x,--indexDir <arg>      Directory with the input indexes
 -y,--year <arg>          Wikipedia year edition (e.g., 2015, 2016...)

Ex: java -cp wikiTailor.v1.0.0.light.jar cat.lump.ir.lucene.Xecutor -l en -y 2015 -i
wikiTailor.ini -n 49024 -d WTlucene/rawFiles -o WTlucene/extraction
```

As before, there are two mandatory arguments that define the Wikipedia edition to use, *-language* and *-year*. The configuration file that specifies the default parameters for the model

is loaded with the *-ini* option. The most relevant parameters for the model can be modified via the command line as seen above. The domain for which you want to extract the articles must be detailed using the *-category* or *-numcategory* options. In the first case, the domain is characterised by the title of a category existing in Wikipedia; in the second case, the associated ID is used. The location to store the Wikipedia edition in raw files, the indexes needed by the Lucene engine and the output directory to store in-domain articles can be specified using *-inputDir*, *-indexDir* and *-outputDir* respectively.

By default, the Xecutor goes through five steps:

Step 1. **DomainKeywords.** Extraction of the vocabulary as a set of lemmas associated to the domain. The *-top* parameter controls the size of the vocabulary (running it with $t = 100$ should be enough). The output of this step is a file with extension *.dict* with the list of lemmas; it used in step 4.

Step 2. **ArticleTextExtractor.** Extraction of all the articles of the Wikipedia edition from the database into plain text in the *-inputDir* directory. The articles can be removed after indexing (step 3).

Step 3. **LuceneIndexerWT.** Indexation of the full collection of articles using the Lucene engine according to its terms (lemmas). The indexes are stored in *-indexDir*.

Step 4. **WikiTailor2Query.** The vocabulary terms of step 1 are used to query the Lucene engine with the documents indexed in step 3. The output file with the list of IDs of the selected articles has extension *.extracted.articles*.

Step 5. **ArticleTextExtractor.** Extraction of the the in-domain articles' contents into plain text files in the *-outputDir*. The list or articles produced in step 4 is needed.

4 Tailoring WikiTailor

4.1 WikiTailor Structure

WikiTailor is organised in four main blocks as shown in the tree distribution of Figure 2. The root folder contains the ant build file *build.xml*, the main configuration file *wikiTailor.ini* and the *README.md* file. The core of WikiTailor is in the *lump* folder and the external libraries are in *thirdparty*. Finally, the configuration files for the first set up are in *configs*.

Only the structure of *lump* is required to add functionalities to WikiTailor. *Lump* contains 7 projects:

lump-aq-basics This project includes several packages to do basic and general operations for text acquisition:

- **algebra** includes classes for operating over vectors and matrices
- **check** includes classes to check whether a given parameter (or any object) is as expected; e.g. it is not empty, not null, true, etc. It includes a few methods to throw errors
- **io-files** includes classes to read and write files in different ways as well as a fool CSV reader and writer
- **log** includes classes and configuration files to keep logs

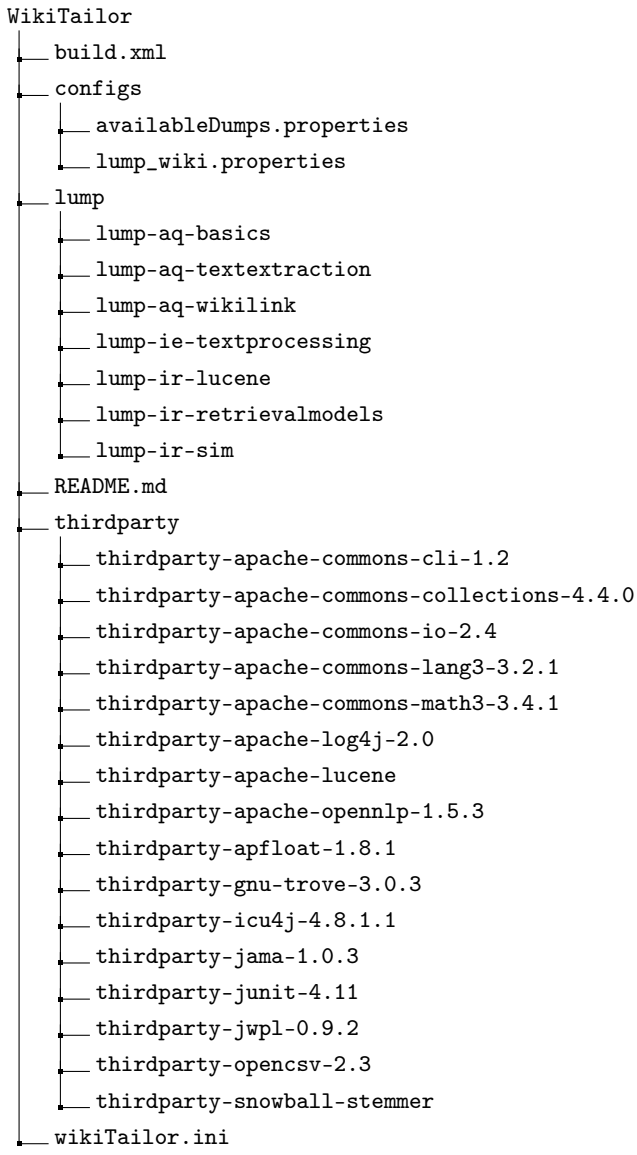

Figure 2: Tree distribution of the WikiTailor package.

structure includes classes with basic data structures used by more specific packages

lump-aq-textextraction This project includes classes to extract and manipulate text, in particular Wikitext from a local preprocessed copy of Wikipedia (through JWPL).

wikipedia includes the classes to explore Wikipedia's category graph and extract the articles according to its domain

lump-aq-wikilink This project includes classes to communicate with the MySQL database: access configuration data, connection settings, and query tools.

config includes the class to access the Wikipedia-db-related configuration

connection includes a class to manage the connection with the MySQL database

jwpl includes methods to initialise the database and the properties file with the language-dependant keywords associated to Wikipedia

lump-ie-textprocessing This project includes classes to extract and manipulate text from files. It includes the abstract classes for decomposing a text and implementations for sentence detection, tokenization, character and word n-gram extraction, and stopwording, among other features.

ner includes the classes to extract named entities with opennlp

ngram contains the classes for extracting both word and character n-grams from a text

sentence has one single class that detects sentences in different languages (currently including only Arabic, German, Greek, English and Portuguese; the English model is used for other languages, such as Spanish).

stopwords contains a class to discard/capture the stopwords in a text. The lists of stopwords (in [..]stopwords/lists) are currently available for Arabic, Basque, Bulgarian, Catalan, Czech, German, Greek, English, French, Occitan, Portuguese, Romanian and Spanish.

word contains a stemmer factory that relies on Snowball and Lucene's stemmers (currently available: Dutch, English, French, German, Italian, Spanish, among others) and a word decomposer based on ICU4J.

lump-ie-lucene This project includes the classes related to extraction of Wikipedia's articles using IR methods, all of them built on top of Lucene.

lump-ie-retrievalmodels This project includes al the general IR-related classes. It includes indexing, querying and both document representation and similarity measures.

document includes classes to represent a document's contents with different characterisations (e.g. bag of words and n-grams)

similarity contains the classes to compute similarity between two document vectors

lump-ie-sim This project includes an interface with the minimum required methods to code a similarity model and several implementations.

ml includes monolingual similarities, currently ESA

cl includes crosslingual similarities, currently a length model and CL-ESA

4.2 Adding a New Language

4.2.1 Graph-based System

Step 1. **Wikipedia Labels.** Include the labels used in wikitext in the new language (e.g. Category, Image, Redirect...) to *cat.lump.aq.wikilink.jwpl.languageConstants.properties*. You will have to look for most of the labels in the wikitext itself. Be especially careful with the Category, Disambiguation and Redirect labels as they are used to consider or discard a Wikipedia document. As seen in the following excerpt of the English set of labels, for some tags there is more than one option; if that is the case they are separated by three commas:

```
#LANGUAGE: ENGLISH  (en)
english_image=File,,,Image
english_category=Category,,,Categories
english_see_also=See also
english_references=References
english_notes=Notes
english_bibliography=Bibliography
english_further_reading=Further reading
english_external_links=External links
english_redirect=#redirect
english_disambiguation=Disamb,,,Disambig,,,disambiguation,,,Geodis,,,Hndis,,,
english_acronym=APPARENTLY NO ACRONYM
english_index=sia,,,index,,,Surname,,,Given name
```

Step 2. **Preprocessing.** Check the alphabet of your language is covered in case it is non-latin in *cat.lump.ie.textprocessing.TextPreprocessor.java*.

Step 3. **Stopwords.** Add a list of stopwords to package *cat.lump.ie.textprocessing.stopwords.lists* and add the new language to *cat.lump.ie.textprocessing.stopwords.Stopwords.java*.

Step 4. **Stemmer.** Provide a Snowball stemmer at package *cat.lump.ie.textprocessing.word* and include it in *StemmerFactory.java*. In case Snowball has not a stemmer for this language, look if it is available in Lucene (currently Lucene 3.5 is used) and include it in *AnalyzerFactoryLucene.java*. If Lucene does not have a stemmer either, you will have to implement your own and call it from *TextPreprocessor.java*.

Step 5. **Acknowledge the language.** Add your language in the list of available languages to *cat.lump.aq.wikilink.Languages*

4.2.2 IR-based System

Step 1. **Analyser.** Add an Analyser for your language. Hopefully it is already implemented in Lucene; then add it to class *cat.lump.lucene.engine.AnalyzerFactory*. Specify the characteristics of your stemmer in case Lucene is not using the Snowball one at package *cat.lump.lucene.engine*.

Step 2. **Tokeniser.** Add an Analyser to act on queries to *cat.lump.lucene.query.LuceneTokenizer*.

Step 3. **Acknowledge the language.** Add your language to the list of available languages in *cat.lump.lucene.index.config.lucene.properties*.

5 Additional Utilities

This section describes some utilities that can be used within WikiTailor, but are not part of its core.

5.1 Term Frequencies

For several purposes, it is useful to have at hand the frequencies of the terms appearing in the articles. Once the articles have been downloaded, these frequencies can be calculated using the *ArticlesTFs* class:

```
user@machine:~/WT-v1.0.0/java -cp wikiTailor.v1.0.0.light.jar
  cat.lump.aq.textextraction.wikipedia.utilities.ArticlesTFs -h

usage: java -cp wikiTailor.v1.0.0.light.jar
       cat.lump.aq.textextraction.wikipedia.utilities.ArticlesTFs [-h] -l <arg> [-p <arg>]

where the arguments are:
 -h,--help              This help
 -l,--language <arg>    Language of interest (e.g., en, es, ca)
 -p,--path2root <arg>   Path to the folder where plain/ is (default: current)

Ex: java -cp wikiTailor.v1.0.0.light.jar
cat.lump.aq.textextraction.wikipedia.utilities.ArticlesTFs -l en -p /home/user/WT/en/
```

The input for this class is the language of the documents, specified via the parameter *-language*; and their location, specified via *-path2root*. The articles must be in raw text format and can be obtained by any means, but they have to be stored in the same way as WikiTailor does. As an example, for the extraction of some English Wikipedia articles, one would obtain a folder with the structure shown in Figure 3.

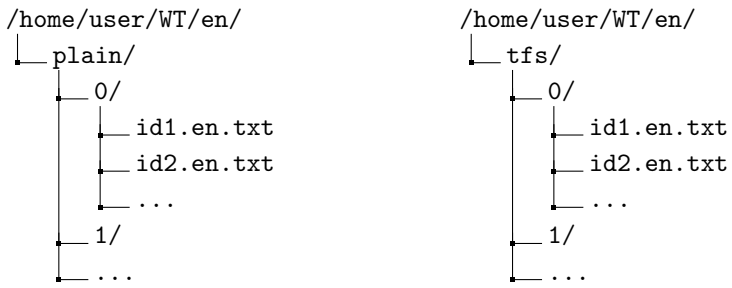

Figure 3: Example of a folder structure with English articles and their associated term frequency files.

Given the path to the folder where *plain/* lies, this class creates a *tfs/* folder with the same structure as the one holding the raw files. There is one file associated to each article and identified by its ID, with a list of terms and the number of times they appear within the document.

5.1.1 Term Definition

no se donde deberia ir esto. Seguramente con el modelo. Within WikiTailor a term is defined as a pre-processed token. We use a standard IR pre-processing:

Step 1. **Normalisation.** Normalisation of punctuation: single and double quotes, dots and dashes.

Step 2. **Tokenisation.** Input sentences are tokenised with the ICU4J framework available at AItools[5]

Step 3. **Lowercasing.**

Step 4. **Punctuation removal.** Removes the following characters: ! “ # $ % & ’ ( ) * + , - . : ; < = > ? @ [\ ] _ ‘ { | } ∼ ˆ .

Step 5. **Stopword removal.** For languages written with alphabets other than Latin, also English stopwords are removed.

Step 6. **Stemming.** The remaining tokens are stemmed using either Snowball or Lucene stemmers.

Step 7. **Diacritic removal.** For languages where stemmers do not perform any diacritic removal it is done afterwards.

Step 8. **Non-alphabetic removal.** Removes tokens written with non-alphabetic characters.

Step 9. **Length constraint.** Removes the tokens with a length less than minimum size. This size is set to 4 for all the languages except Arabic, where we use 3 to keep all triliteral roots.

5.2 Multilingual Title Extraction

As a first step towards the extraction of parallel fragments from Wikipedia's articles, we make available a simple class to extract the titles of any content document across all the languages in which a document exists. This set of titles are parallel in all the languages, and belong to a concrete domain if the selected articles have been extracted with WikiTailor for that domain.

The class *cat.lump.aq.textextraction.wikipedia.utilities.CommonNamespaceFinder* identifies the common articles (namespace=0) or categories (namespace=14) across n languages in Wikipedia departing from n ID files generated in Step 6 of the graph-based system (Section 3.2.1) or in Step 4 of the IR-based system (Section 3.2.2). To use it do:

```
user@machine:~/WT-v1.0.0/java -cp wikiTailor.v1.0.0.light.jar
   cat.lump.aq.textextraction.wikipedia.utilities.CommonNamespaceFinder -h

usage: java -cp wikiTailor.v1.0.0.light.jar
       cat.lump.aq.textextraction.wikipedia.utilities.CommonNamespaceFinder
       -f <arg>  [-h]  [-o <arg>]  -y <arg>

where the arguments are:
 -f,--files <arg>     List of files with the IDs of the articles/categories
                      (e.g. ca.591034.4.articles en.691182.5.articles)
                      The first two letters must indicate the language.
 -h,--help            This help
 -o,--outpath <arg>   Optional: save the output into this directory (default: current)
 -y,--year <arg>      Wikipedia year edition (2013, 2015, 2016)

Ex: java -cp wikiTailor.v1.0.0.light.jar
    cat.lump.aq.textextraction.wikipedia.utilities.CommonNamespaceFinder -y 2015
```

[5] http://www.uni-weimar.de/en/media/chairs/webis/research/selected-projects/aitools/

```
    -f ca.591034.4.articles en.691182.5.articles es.1733769.4.articles

Ex: java -cp wikiTailor.v1.0.0.light.jar
    cat.lump.aq.textextraction.wikipedia.utilities.CommonNamespaceFinder -y 2015
    -f ca.591034.4.category en.691182.5.category es.1733769.4.category
```

Almost all the information to execute the command is obtained from the name of the ID files, so do not modify WikiTailor output's files nomenclature. The first two characters of these files indicate the language of the Wikipedia edition, and the extension marks the Wikipedia namespace[6] to consider. Given this input, one gets two files with parallel titles for every namespace, one corresponding to the union of articles and the other one to the intersection. With the union, one obtains a larger corpus and coverage; with the intersection, one obtains a smaller corpus but with titles that belong more precisely to the desired domain.

Union. Looks for the articles that appear in any of the selected languages and builds a set with its union. So, for languages other than the wider one which is taken as the guide, if the articles exist in the DB but do not appear in the ID files, they are added.

Intersection. Looks for the articles that appear in all the selected languages simultaneously. If they exist in the DB but do not appear in the files with the IDs, the articles are discarded.

The name of the output files describe the languages, categories, namespace and method used. Each line of the files contains the ID and the title of a document for every language. Blank spaces in the titles are substituted by underscores. Example:

```
user@machine:~$ head en.691182.es.1733769.t0.union

1126536 Optimization_problem    26644 Optimización_(matemática)
5632946 Shift_work_sleep_disorder     4124672 Desorden_de_sueño_por_turno_de_trabajo
1126777 Nightbreed    5003757 Nightbreed
```

6 Towards WikiParalel

References

[1] España-Bonet, C., Barrón-Cedeño, A., and Màrquez, L. Tailoring Wikipedia for in-Domain Comparable Corpora Extraction. *In preparation.*

[2] Zesch, T., Müller, C., and Gurevych, I. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the 6th International Conference on Language Resources and Evaluation* (Marrakech, Morocco, May 2008). electronic proceedings.

[6] https://en.wikipedia.org/wiki/Wikipedia:Namespace

A Default Values for Wikipedia Pre-Processing

This appendix lists the names of the category pages needed for running the JWPL pre-processing (Section 2.2) for the languages currently available in WikiTailor.

Mind that the names of the main category (Table A.1) or the category marking disambiguation (Table A.2) pages may change over time. E.g. the English category for disambiguation pages was called "Disambiguation" for a long time, while now it is "All_disambiguation_pages". Labels here shown have been obtained in March 2016, but can be modified in the future. You can check their validity or the value for other languages at:

`https://www.wikidata.org/wiki/Q1281` for content pages, and
`https://www.wikidata.org/wiki/Q1982926` for disambiguation pages.

A.1 Wikipedia Main Category Name

Language	LAN	MAIN_CATEGORY_NAME
Arabic	ar	تصنيف: محتويات ويكيبيديا
Basque	eu	Kategoria:Edukiak
Bulgarian	bg	Категория:Начална категория
Catalan	ca	Categoria:Principal
Czech	cs	Kategorie:Kategorie
English	en	Category:Contents
French	fr	Catégorie:Accueil
German	de	Kategorie:!Hauptkategorie
Greek	el	Κατηγορία:Κατηγορίες
Occitan	oc	Categoria:Acuèlh
Portuguese	pt	Categoria:Conteúdo
Romanian	ro	Categorie:Pagina principală
Spanish	es	Categoría:Índice de categorías

all this empty space...

A.2 Wikipedia Disambiguation Category Name

Language	LAN	DISAMBIGUATION_CATEGORY_NAME
Arabic	ar	تصنيف: صفحات _توضيح
Basque	eu	Kategoria:Wikipediako argipen orriak
Bulgarian	bg	Категория:Пояснителни страници
Catalan	ca	Categoria:Pàgines de desambiguació
Czech	cs	Kategorie:Rozcestníky
English	en	Category:Disambiguation pages
French	fr	Catégorie:Homonymie
German	de	Kategorie:Begriffsklärung
Greek	el	Κατηγορία:Αποσαφήνιση
Occitan	oc	Categoria:Omonimia
Portuguese	pt	Categoria:Desambiguação
Romanian	ro	Categorie:Dezambiguizare
Spanish	es	Categoría:Wikipedia:Desambiguación