WikiTailor

Technical Manual

Version 1.0

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

WORKING DOCUMENT

February 20, 2016

Abstract

WikiTailor is a Java toolkit designed to extract and analise corpora from Wikipedia in any language and domain[1]. 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). quieres incluir al QCRI?

Contents

1 Introduction

2 Prerequisites and Installation

This software is distributed as a stand-alone jar[2], so nothing else but a Java Virtual Machine is needed. The source code is also available[3] in which case, it can be compiled using the included Ant buildfile (see Section 2.3). In both cases a Wikipedia dump of the desired languages must be stored in a database to retrieve the articles off-line. Sections 2.1 and 2.2 describe how to download the data and create the database in a format compatible with WikiTailor.

2.1 Downloading the Dumps

In order to preprocess the Wikipedia articles a dump of the desired language(s) must be downloaded from the Wikimedia webpage:

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

where, and from here on, `LAN` is a two-character language identifier with the ISO 639 codes[4] and `DATE` is the date of the dump in the `YEARMONTHDAY` format.

The dump consists of several sql tables. WikiTailor needs the general purpose tables:

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

and three additional tables to relate articles in different languages and follow redirections:

`[LAN]wiki-[DATE]-langlinks.sql.gz`
`[LAN]wiki-[DATE]-page.sql.gz`
`[LAN]wiki-[DATE]-redirect.sql.gz`

The first group can be preprocessed and upload into a MySQL database using the JWPL DataMachine; the second group can be uploaded directly.

2.2 The JWPL DataMachine and the Database

JWPL (Java Wikipedia Library) is a Java-based application programming interface that allows to access all information in Wikipedia [2]. WikiTailor uses its DataMachine to create a MySQL database with a preprocessed Wikipedia dump and as a external library to parse the MediaWiki syntax.
The full process to deal with the dumps can be summarised in the following steps:

Step 1. Download the JWPL DataMachine and the tables.sql file from
`https://dkpro.github.io/dkpro-jwpl/`

Step 2. Database setup. Create a MySQL database (use UTF-8 encoding) and afterwards the tables:

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

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

[2] `http://cristinae.github.io/WikiTailor/dwnld/wikiTailor-v1.0.0.with-dependencies.tar.gz`
[3] `https://github.com/cristinae/WikiTailor`
[4] `http://www.loc.gov/standards/iso639-2/php/code_list.php`. ISO 639-1 is used when available.

where a DBname in the format [DBrootname]_wiki_[LAN]_[YEAR] is expected from WikiTailor.

Step 3. JWPL preprocessing. 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[5],
DISAMBIGUATION_CATEGORY_NAME is the name of the category that contains the disambiguation categories[6] and,
SOURCE_DIRECTORY is the path to the directory containing the data dumps.

This should create a lot of new data files in a "output" subfolder of each input folder.

Mind that the names of the main category or the category marking disambiguation 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".

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 تصنيف:محتويات_ويكيبيديا تصنيف:صفحات_توضيح ./ar/
```

Step 4. 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:

[5] A complete list with the labels for the main category can be obtained in https://www.wikidata.org/wiki/Q1281.

[6] A complete list with the labels for the disambiguation pages can be obtained in https://www.wikidata.org/wiki/Q1982926.

```
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 Installation

This subsection provides the basic instructions for installing WikiTailor in case you decide not to use the compiled jar. All the following steps work on a linux distribution. Be sure that you have Git, Apache Ant and Java (version > Java 7) installed, otherwise use your favourite package manager or do:

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

2.3.1 Building WikiTailor

First of all download the last version of the code. Clone WikiTailor from the repository:

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

cd into the WikiTailor folder and run ant to build the source:

```
ant deploy-src
```

Run the basic tests to check everything is fine:

```
ant testNoDB
```

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 uses 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

For the current version, we have included the external libraries in the git repository itself, so when you build the source code the needed libraries are already added. Ahora hay el paquete light y el completo, solo se deberia compilar uno por default aunque demos los dos como binarios. Cual?

3 Description and Usage

paragraph description

3.1 Initial Setup

the ini file

```
################################################
###   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 systems available in WikiTailor can be executed from beginning to end by using an *Xecutor* class implemented within their corresponding packages[7]. In both cases, each subtask of the system can be also done individually either by limiting the required steps to the Xecutor using command line arguments *-start* and *-end*, or by using the specific class for the task. In the following, we describe how the full pipeline for each system works.

3.2.1 The Graph-based System

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

```
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*, and 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 goes through seven steps:

Step 1. **Runs 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

[7] We recommend extracting the articles with the graph-based method since it has been shown to produce better results [1].

$t = 100$ should be enough). The output of this step is a file with extension *.dict* with the list of lemmas; it used in step 3.

Step 2. **Runs CategoryExtractor.** Identification of the top 50 levels (if existing) 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`
The information is further used in steps 3 and 4.

Step 3. **Runs CategoryNameStats.** Estimates 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, that is, belonging to the domain, are stored in a file with extension *.stats*.

Step 4. **Runs CategoryDepth.** Searches in the category tree for the depth up to which articles belong to the domain. This is defined by the parameter *-model* that is 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. **Runs createCategoriesFile.** Extracts 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. **Runs ArticleSelector.** Extracts 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. **Runs ArticleTextExtractor.** Extracts the in-domain articles' contents into plain text files. The list or articles produced in step 6 is needed.

3.2.2 The IR-based System

Similarly to 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*, and 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. **Runs 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. **Runs ArticleTextExtractor.** Downloads all the articles of the Wikipedia edition defined by (*-language*, *-year*) from the database into plain text in the *-inputDir* directory. The articles can be removed after indexing (step 3).

Step 3. **Runs LuceneIndexerWT.** Indexes the collection of articles of a full Wikipedia edition using the Lucene engine according to its terms, that is, lemmas. The indexes are stored in *-indexDir*.

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

Step 5. **Runs ArticleTextExtractor.** Extracts 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 Architecture

`WikiTailor`

```
├── build.xml
├── configs
│   ├── availableDumps.properties
│   └── lump_wiki.properties
├── lump
│   ├── lump-aq-basics
│   ├── lump-aq-textalignment
│   ├── lump-aq-textextraction
│   ├── lump-aq-wikilink
│   ├── lump-ie-textprocessing
│   ├── lump-ir-lucene
│   ├── lump-ir-retrievalmodels
│   └── lump-ir-sim
├── README.md
├── thirdparty
│   ├── thirdparty-apache-commons-cli-1.2
│   ├── thirdparty-apache-commons-collections-4.4.0
│   ├── thirdparty-apache-commons-io-2.4
│   ├── thirdparty-apache-commons-lang3-3.2.1
│   ├── thirdparty-apache-commons-math3-3.4.1
│   ├── thirdparty-apache-log4j-2.0
│   ├── thirdparty-apache-lucene
│   ├── thirdparty-apache-opennlp-1.5.3
│   ├── thirdparty-apfloat-1.8.1
│   ├── thirdparty-gnu-trove-3.0.3
│   ├── thirdparty-icu4j-4.8.1.1
│   ├── thirdparty-jama-1.0.3
│   ├── thirdparty-junit-4.11
│   ├── thirdparty-jwpl-0.9.2
│   ├── thirdparty-opencsv-2.3
│   └── thirdparty-snowball-stemmer
└── wikiTailor.ini
```

4.2 Including a new Language

/home/cristinae/pln/workspace/lump2/lump2-aq-textextraction/HOWTO.txt

5 Evaluating the Extractions

6 Additional Utilities

7 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.