Technical Summary — Selection and Preprocessing of the WeScience Corpus *

Gisle Ytrestøl
University of Oslo, Department of Informatics
gisley@ifi.uio.no

July 10, 2009

1 The WeScience Corpus

Considering the advances in research on syntactic and semantic analysis that has been reached through linguistically annotated corpora, there exist relatively few large-scale linguistic treebanks. More than anything, this can be seen through the dominant role of Penn Treebank (PTB; Marcus et al. (1993)) in data-driven natural language processing (NLP) for English. This corpus consists primarily of Wall Street Journal articles from the late 1980s, and both its subject matter, genre, design decisions and age can make the use if this treebank undesirable for many tasks. Generally speaking, richly annotated treebanks that exemplify a variety of domains and genres (and of course languages other than English) are however not yet available. And neither are broadly accepted gold-standard representations that adequately support a range of distinct NLP tasks and techniques.

In response to a growing interest in so-called eScience applications of NLP—computationally intensive, large-scale text processing to advance research and education—a lot of current research targets scholarly literature, often in molecular biology or chemistry (Tateisi et al. (2005); Rupp et al. (2007); inter alios). Due to the specialized nature of these domains, however, many NLP research teams—without in-depth knowledge of the subject area—report difficulties in actually "making sense" of their data. To make eScience more practical (and affordable for smaller teams), we propose a simple technique of compiling and annotating domain-specific corpora of

*I would like to thank Stephan Oepen for his help and advises in the process of making and preprocessing this corpus, and Dan Flickinger for his work on treebanking this corpus, and for his productive comments and evaluation of the quality of the corpus. This technical summary quotes from *Extracting and Annotating Wikipedia Sub-Domains*, (Ytrestol, Flickinger, and Oepen, 2009).

scholarly literature, initially drawing predominantly on encyclopedic texts from the community resource Wikipedia.[1]

Adapting the Redwoods grammar-based annotation approach (Oepen et al., 2004) to this task, we are constructing and distributing a new treebank of texts in our own field, Computational Linguistics, annotated with both syntactic and (propositional) semantic information—dubbed the WeScience Treebank. Should this approach prove feasible and sufficiently cost-effective, we expect that it can be adapted to additional document collections and genres, ideally giving rise—over time—to an increased repository of "community treebanks", as well as greater flexibility in terms of gold-standard representations.

1.1 Initial Preparations

All articles in the WeScience Corpus are taken from a Wikipedia snapshot released July 2008, and can be downloaded at `http://www.delph-in.net/wescience/enwiki-20080727-pages-articles.xml.bz2`. More recent dumps are available at `http://download.wikimedia.org/enwiki/`.

From this collection, I set up an offline Wikipedia reader on a local computer. As Wikipedia articles are continuously changing, it was important to have this collection offline in order to have a data set that does not change overnight. The offline reader allows the user to browse and search for Wikipedia articles from one specific Wikipedia collection on a local web server. A simple introduction on how to build an offline Wikipedia reader can be found on `http://users.softlab.ece.ntua.gr/~ttsiod/buildWikipediaOffline.html`. Alternatively, one can use MediaWiki[2]. The scripts I have used would however need to be rewritten in order to work with MediaWiki.

2 The Software and Scripts used by WeScience

The following scripts and software tools are used by WeScience (sorted by the order in which they are used):

- `linkextractor.py`
- `makecorpus.py`
- `ccp.py`
- `tokenizer` — `http://www.cis.uni-muenchen.de/~wastl/misc/`

[1] In 2007 and 2008, the interest in Wikipedia content for NLP research has seen a lively increase; see `http://www.mkbergman.com/?p=417` for an overview of recent Wikipedia-based R&D, most from a Semantic Web point of view.

[2] `http://www.mediawiki.org/wiki/MediaWiki`

- `cccp.py`

The Python scripts are available through SVN, `\$LOGONROOT/uio/wescience/bin` in the LOGON repository. They are however not made for large-scale extraction and corpus preparation, and the user may experience that they are inefficient if they are used with a significantly larger number of articles than what is used by WeScience.

3 Domain-Specific Selection of Text

Our goal for the WeScience Corpus is to extract a sub-domain corpus, targeting our own research field—NLP. To approximate the notion of a specific sub-domain in Wikipedia (or potentially other hyper-linked electronic texts), we start from the Wikipedia category system—an optional facility to associate articles with one or more labels drawn from a hierarchy of (user-supplied) categories. The category system, however, is immature and appears far less carefully maintained than the articles proper. Hence, by itself, it would yield a relatively poor demarcation of a specific subject area. For our purposes, we chose the category *Computational Linguistics* and all its sub-categories—which include, among others, *Natural Language Processing*, *Data Mining* and *Machine Translation*—to activate an initial seed of potentially relevant articles. Altogether, 355 articles are categorized under *Computational Linguistics* or any of its sub-categories. These seed articles can be inspected in Appendix B.1, and is found in the LOGON repository (`\$LOGONROOT/uio/wescience/etc`). However, some of these articles seemed somewhat out-of-domain (see Section 3.1 for examples), and several are so-called stub articles or very specific and short, e.g. articles about individual software tools or companies. To compensate for this, we applied a simple link analysis using the script `linkextractor.py`, as described below.

3.1 Link Extractor

It is apparent that many relevant articles are not (yet) associated with either *Computational Linguistics* or any of its sub-categories. To compensate for the limitations in the Wikipedia category system, we applied a simple link analysis and counted the number of cross-references to other Wikipedia articles from our initial seed set. By filtering out articles with a comparatively low number of cross-references, we aim to quantify the significance (of all candidate articles) to our domain, expecting to improve both the recall and precision of sub-domain extraction.

Among the articles that were filtered out from our original set of seed articles, we find examples like *AOLbyPhone* (1 reference) and *Computational Humor* (2 reference). New articles, differently categorized, were activated based on this approach. These include quite prominent examples

like *Machine learning* (35 references), *Artificial intelligence* (34 references) and *Linguistics* (25 references). Of the 355 seed articles, only 30 articles remain in the final selection. Confirming our expectations, filtering based on link analysis eliminated the majority of very narrowly construed articles, e.g. specific tools and enterprises.

However, our link analysis and cross-reference metric also activates a few dubious articles (in terms of the target sub-domain), for example *United States* (9 references). We have deliberately set up our sub-domain extraction approach as a fully automated procedure so far, avoiding any elements of subjective judgment.[3]

The file containing the seed articles was used as input by the Python script `linkextractor.py`. The script extracts the links in the Wikipedia articles and sorts them by frequency. Links containing ":" or "#" are removed, as these links refer to external sources or sub-sections within a Wikipedia article. Figure 1 shows a snippet of the frequency count, after all the links have been extracted from the articles.

```
(...)
cleanLink= {}
for link in allLinks:
    if not ':' in link:
        if not '#' in link:
            if link in cleanLink:
                cleanLink[link] = cleanLink[link] + 1
            else:
                cleanLink[link] = 1
sortedDict =  sorted(cleanLink.iteritems(),
    key=lambda (k,v):(v,k), reverse=True)
for link in sortedDict:
    print link
```

Figure 1: The algorithm for counting and sorting all the links from all the articles, when all the links are stored in the list *allLinks* .

It should however be noted that the desired consistency is not always kept when various articles are being referred to. Thus, we find references to

[3]Except for the removal of the articles "2005" and "2006", which primarily contain lists of occurrences and deaths that took place that year, and not running text.

("/article/Part_of_speech/", 8), as well as ("/article/Parts-of-speech/", 2) – both of these links will lead to a redirect page to the article *Lexical Category*. Although it is to some extent possible to compensate for some of these inconsistencies, for practical reasons we decided on using this initial link analysis. Please see Appendix C for a more extensive discussion on this matter.
Usage:

```
python linkextractor.py seedArticleLinks.txt >
linksSortedByFreq.txt
```

`linkextractor.py` takes a list of links to Wikipedia articles as parameter, and prints the number of links (e.g. Wikipedia articles) in the articles as standard output, sorted by frequency. For the WeScience Corpus, we used all articles that had eight cross-references or more, see Appendix B.2, 112 links in total.

3.2 Article Extraction

The Python script `makecorpus.py` takes a list of Wikipedia links as parameter, and will try to store these articles to the local disk. The script will only work with the Wikipedia offline reader described in Section 2. The paths to the Wikipedia source file, as well as the path to where the articles will be stored, are hard coded into the script, and may be changed in an alternative configuration. By default, a copy of the Wikipedia source article will be stored to `/var/tmp/result` when a given Wikipedia article is opened in the offline reader. This file will be copied to the path provided in `newfilepath` in the function `copyCorpus` in the script.

The script detects if the pages visited contains a redirect link, and will follow the redirect link no more than five time—if it does not reach a Wikipedia article by then, it will move on to the next link. To suppress the linguistically less rewarding stub articles, we further applied a minimum length threshold (of 2,000 characters, including markup) and were left with 100 Wikipedia articles and approximately 270,000 tokens.

Usage: `python makecorpus.py articlelinks`

The articles will be stored with a three digit identifier prefix, followed by the name of the article and the suffix `source`, the result for the WeScience Corpus can be inspected in Appendix B.3.

4 Wikipedia Markup

Wikipedia articles are edited in *Wiki Markup Syntax*, a straightforward logical markup language that facilitates on-line rendering (as HTML) for display

in a web browser. Again, there are Wikipedia guidelines and conventions for how to edit or add content, aiming to keep the architecture and design as consistent as possible. In preparing the WeScience Corpus we aim to strike a practical balance between (a) preserving all linguistic content, including potentially relevant markup, and (b) presenting the corpus in a form that is easily accessible to both humans and NLP tools. From the raw source files of Wikipedia articles, we eliminate markup which is linguistically irrelevant—some meta information or in-text image data, for example—but aim to preserve all markup that may eventually be important for linguistic analysis. Markup indicating bulleted lists, (sub)headings, hyper-links, or specific font properties, for example, may signal specialized syntax or use–mention contrasts.

Example (1) and (2) are two examples of Wikipedia markup that we want to preserve, because it closely interacts with "core" linguistic content:

(1) [10120240] |* Design of [[parser]]s or [[phrase chunking—chunkers]] for [[natural language]]s

(2) [10621290] |For example, in the following example, "one" can stand in for "new car".

The WeScience Corpus provides gold-standard "sentence" boundaries (sometimes sentential units are not sentences in the linguistic sense) with unique sentence identifiers. Examples (1) and (2) show the actual WeScience file format, where each sentence is prefixed by its identifier and the "|" separator symbol. In (1), the initial * indicates items in a bulleted list (which can exhibit various specialized syntactic patterns) and the square brackets show Wikipedia hyper-links. Example (2) on the other hand, shows the use of *italics* (the interpretation of the double apostrophe in Wikipedia markup) for the purpose of quoting; i.e. the use citation–mention distinction is made as a font property only. Section 4.2 will give an in-depth discussion on the tools used in this pre-processing stage.

4.1 Wikipedia Templates

In order to maintain consistency within Wikipedia articles, a number of templates are used to display content that should be identical throughout (English) Wikipedia. In Wiki Markup Syntax, templates appear within curly brackets.

There are two ways of using templates in Wikipedia: *transclusion* and *substitution*.

> The former will include the content of {{Template Name}} on the fly whenever the article is loaded, while the latter will permanently insert the content of the template into the article. With substitution, even if the template content is modified at

a later date, the article's content will not change. The common method for using template messages is transclusion, implemented by usage of a tag with the form {{template name}}, in whatever article or talk page one wants the template code/text to be shown. (`http://en.wikipedia.org/wiki/MediaWiki_custom_messages`)

For transclusion, a text box is typically inserted in the article. E.g. the template {{Disputeabout|"'The topic of dispute"'}} within the article will insert the text box in Figure 2 on top of the article.

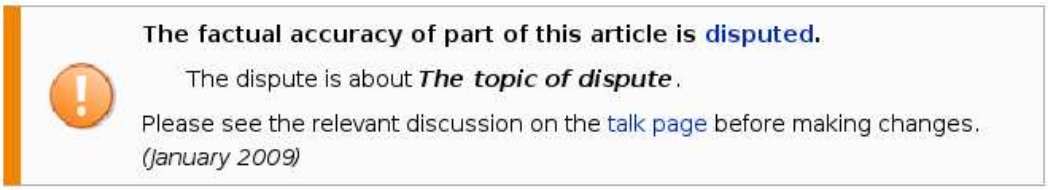

Figure 2: The text box inserted by the *Disputeabout* template.

Adding *subst:* in front of the template will permanently replace the template with the text of the template, and this is not an issue we have to deal with in the WeScience Corpus.

A small proportion of the templates contains content that will appear within the running text in the Wikipedia articles. E.g. the template *Fact* will be rendered [*citation needed*] in the browser. As the templates are defined globally, they are stored in a database, and retrieved each time a Wikipedia page is read. By default, templates are removed from the WeScience Corpus, as they contribute little to the linguistic content of the sentence.

However, we have a few custom-made rules for some of the more frequent templates that are important for the sentences they occur in. An example on this is the template *IPA*, that makes sure the IPA transcription is correctly rendered by the browser. The IPA template

(3) {{IPA|/'kærəktəz/}}

will be converted to the following HTML code in Wikipedia:

(4) `<span title="Representation in the International Phonetic Alphabet (IPA)" class="IPA">/'kærəktəz/</span>`[4]

In WeScience, the IPA templates are preserved when they occur in running text, as in example (5).

(5) [10421470] —E.g.: "buono" {{IPA|['bwɔno]}}, "ieri" {{IPA|['jɛri]}}.

[4] `http://en.wikipedia.org/wiki/Template:IPA`

Another example is the {{lang}} template, used for indicating that a given span of text belongs to a particular language.

(6) [10302260] |: "{{lang|de|Der alte Mann gibt mir das Buch heute.}}"

The use of templates are quite inconsistent in many articles, and probably dependent on the person editing the text (as one is free to introduce own templates). The result of this seems to be that the use of templates vary according to the author of the article/section in Wikipedia. E.g. within the *Japanese Language* article there is no consistency as to whether passages and words in Japanese are written with the {{Nihongo}} template[5], the {{lang|jap}} template, or no template at all.

4.2 Markup Removal

Unwanted markup and entire sections from the source articles have been automatically removed, mainly by the use of regular expressions in the `Corpus Clean` pipeline (see Section 4.2.1). Removed parts of the articles are, as we see it, irrelevant to linguistic analysis, including entire sections—like *See Also*, *References*, or *Bibliography*—or links to images, comments made by other users, templates (as described in Section 4.1) and various Wikipedia-internal elements.

Once reduced to what we consider (potentially) relevant linguistic content, we applied semi-automated sentence segmentation. In a first, automated step, all linebreaks were removed from the original source text, and the open-source package `tokenizer` was used to insert sentence boundaries. This is a rule-based tool which proved very capable as a sentence segmenter. The procedure was further optimized by some customization, based on a manual error analysis. Most of the errors made by the segmenter could be attributed to "misleading" (remaining) markup in its input (`tokenizer`, by default, expects "pure" text). For instance, the tool initially failed to insert segment boundaries between some numbered list elements (where the Wikipedia markup "#", in a sense, takes on the function of sentence-initial punctuation). To improve the quality of the sentence segmentation, we would either

(7) a. improve the cleansing of the Wiki Source Markup, so that `tokenizer` would process as "normal" text as possible, or
b. force segment boundaries in cases where `tokenizer` was known to fail.

Cases as in (7-a) are discussed in Section 4.2.1, and cases similar to (7-b) are discussed in Section 4.3.

[5] *Nihongo* means *Japanese*.

A second round of error analysis on a sub-set of 1,000 segments suggests a residual error rate of about 4 per cent, with half of these text preparation errors due to incomplete handling of wiki mark-up (e.g. for `<math>` and `<code>` blocks, colons marking indentation, and some hyperlinks). The other half of these errors are due to missing or spurious sentence breaks (often due to unusual punctuation clusters), and to confusion of picture captions or section headers with main text. The remaining errors after the pre-processing stage will corrected manually in the treebanking process.

4.2.1 Corpus Clean

`Wiki Markup Syntax` $\Longrightarrow$ `ccp.py` $\Longrightarrow$ `tokenizer` $\Longrightarrow$ `cccp.py` $\Longrightarrow$ `WeScience Format`

Figure 3: `Corpus Clean` pipeline

The articles are cleansed using the pipeline `Corpus Clean`, where `tokenizer` is executed on the output of the Python script `ccp.py`, and `tokenizer`'s output serves as input for `cccp.py`. The pipeline should remove undesirable parts from the source text, insert sentence segments and equip each sentence with a unique identifier. The cleansing process is mostly based on a number of regular expressions. These Python scripts remove amongst others content in curly brackets, images, galleries and HTML code, and entire sections like *See also*, *References*, *Bibliography* etc. Figure 4 and 5 show an example of only a few of the regular expressions that were used to strip the articles for unwanted content, markup and metadata.

Note that the regular expressions `regrelated` and `regexternal` remove the rest of the entire article. When either of these two sections appear in a Wikipedia article that follows the conventions, we know that the rest of the article should contain mainly lists, references and other elements which will not serve as good input sentences for our parser and it is therefore removed. However, there are some articles where the Wikipedia conventions are not followed, and a section name is used multiple times. The script therefore uses so-called look-aheads to make sure, given that a section name appears multiple times, only the last section in the article is removed.

The regular expressions `regcurley1` and `reggallery` will match anything that appears inside either curly brackets or `<gallery>`...`</gallery>`, and remove the entire content within these.

To convert the cleansed output to a one-sentence-per-line structure, all linebreaks were removed from the original source text. We then ran the sentence boundary detector `tokenizer`, and used this to format the text according to the desired structure. After these stages, each separate article is written to a new file without any manual editing.

```
regcurly1 = re.compile(r'(\({{[^}^{]*?}}\)|
\'*{{[^}^{]*?}}\'*)',re.MULTILINE | re.DOTALL)
regwikitable1 = re.compile(r'{[^}]*?class="?wikitable"?[^}]*?}',
re.MULTILINE | re.DOTALL)
regwikitable2 = re.compile(r'[^{]{\|[^\\][^}^{]*?}',
re.MULTILINE | re.DOTALL)
reggallery = re.compile(r'<gallery.*?</gallery\s?>',
re.MULTILINE | re.DOTALL)
regsourcelookahead = re.compile(r'==+\s?
(?=Sources.*?==+\s?Sources)', re.MULTILINE | re.DOTALL)
regsourcelookaheadrestore = re.compile(r'___(Sources)(.*?==+)',
re.MULTILINE | re.DOTALL)
regrelated = re.compile(r'==+\s?Related web sites\s?.*',
re.MULTILINE | re.DOTALL)
regexternal = re.compile(r'==+\s?External links\s?.*',
re.MULTILINE | re.DOTALL)
```

Figure 4: Regular expressions from the pipeline `Corpus Clean` used to remove unwanted content from the Wikipedia articles.

4.3 Sentence Segmentation

We tested a few different sentence boundary (SB) detectors before deciding which to use. A natural choice would have been to use the NTLK (Natural Language Toolkit[6]) open source Python modules which includes a SB detector. We did however not find its performance sufficiently good, and decided to use `tokenizer` instead. This SB detector is used in a pipeline between stage one and stage two of the Python scripts that remove and convert the articles from `Wiki Markup Syntax` to the desired format.

Since `tokenizer` is designed to be used on normal text, its performance suffered due to the frequent use of markup and non-standard formatting in the article. To compensate for this, we made additional regular expression to be used where `tokenizer` was likely to fail. In example (1) and (2), *End of Line* XML codes are inserted between elements in bulleted list, as sentences in these list often lack a full stop at the end of the sentence. The `<EOS />` tags are later converted to line breaks.

Alternatively, there are markup elements that should never span segment boundaries (e.g. Wikipedia links, `<source>` and `<code>` blocks). In the preprocessing, End-Of-Sentence-Tags inserted by `tokenizer` are removed after the segmentizer stage using temporarily placeholders and regular expression, see Figure 7.

[6] `http://nltk.org`

```
def regCleanFile(self,input):

        input = regex.regcurly1.sub('',input)
        input = regex.regwikitable1.sub('',input)
        input = regex.regwikitable2.sub('',input)
        input = regex.reggallery.sub('',input)
        (...)
        return input

def removeEnd(self,input):
        input = regex.regsourcelookahead.sub(r'___',input)
        input = regex.regsourcelookahead.sub(r'___',input)
        (...)
        input = regex.regsources.sub('',input)
        input = regex.regsourcelookaheadrestore.sub(r'\2\1\2',input)
        (...)
        input = regex.regrelated.sub('',input)
        input = regex.regexternal.sub('',input)
        (...)
        return input
```

Figure 5: Removal of unwanted markup and content from Wikipedia articles.

4.3.1 Usage

The scripts are used in a pipeline with `tokenizer`. `ccp.py` takes the source folder as input parameter (`-i`), and writes the entire corpus to one single file. The option `-n` means "no redirect processing", and should be used if the articles have been extracted by the script `makecorpus.py`, since this script makes sure none of the links lead to a redirect page. `tokenizer` should be invoked as indicated in Figure 8. The last script in the pipeline, `cccp.py`, takes the output from tokenizer as input, and writes the output to a folder specified by the user. The optional parameter `-l` specifies the maximum number of lines in each output WeScience files ("sections" of a sort), for the WeScience Treebank each section comprises up to 1,000 sentence, and no article is split between two files. If no maximum number of lines is specified, the entire corpus will be dumped to one single file. Figure 8 demonstrates the pipeline in use, and can be run as a shell script.

From `regex.py`:

```
regbullets = re.compile(r'(^[#*].*?)\n', re.MULTILINE |
  re.DOTALL)
regindentcolon = re.compile(r'(^[;:][:;]?.*?)\n',
  re.MULTILINE | re.DOTALL)
regbullets2 = re.compile(r'(^\*.*?)\n', re.MULTILINE |
  re.DOTALL)
```

From `regCleanFile(self,input)` in `ccp.py`:

```
input = regex.regbullets.sub(r'<EOS />\1<EOS />', input)
input = regex.regbullets2.sub(r'<EOS />\1<EOS />', input)
input = regex.regindentcolon.sub(r'<EOS />\1<EOS />', input)
```

Figure 6: Insertion of End-of-Sentence Tags in lists.

4.4 The WeScience Format

In the tradition of the Un*x operating system, we have opted for a textual, line-oriented exchange format for the WeScience Corpus. The central unit of analysis in our setup is the sentence, so it is convenient to structure the corpus around this. Each sentence starts with an identifier—this is a eight-number digit that always starts with 1. The next three number specify the article number it is derived from, and the second three refer to the line number. The last number is initially 0, this final slot is reserved for any later corrections and changes on the original line.

Sentence number 49 in article 65 has thus the following identifier:

Placeholder	Article Number	Sentence number	Decimal
1	065	049	0

Table 1: Identifier for sentence (8).

(8) [10650490] |These rules can be formally expressed with [[attribute grammar]]s.

This line-oriented format follows the traditions of the Redwood corpus (Oepen et al., 2004).

From `regex.py`:

```
regremoveeosincode = re.compile
  (r'(<code[^<]*?)<EOS />(.*<EOS />)*?(.*?</code>)',
  re.MULTILINE | re.DOTALL)
```

From `cccp.py`:

```
def removeEosInCode(self,input):
    while regex.regremoveeosincode2.search(input):
        hit = regex.regremoveeosincode2.search(input)
        span = hit.span()
        removedeos = regex.regeos.sub(r'',input[span[0]:span[1]])
        removedeos = removedeos.replace("<code","<___code")
        removedeos = removedeos.replace("</code","</___code")
        input = input[:span[0]]+removedeos+input[span[1]:]
    return input

def processFileAfterToken(self,input):
    (...)
        input = self.removeEosInCode(input)
        (...)
        input = input.replace("<___","<")
        input = input.replace("</___","</")
        (...)
        return input
```

Figure 7: Removal of End-of-Sentence Tags in code elements.

5 Further Work

In the hope that our WeScience efforts may stimulate adaptation by others, we have released a first version in early 2009. This initial release of the corpus and partial treebank is available on the WeScience home page: `http://www.delph-in.net/wescience/`. By June 30th, eight sections (of 16) are treebanked and can be downloaded, this number will increase in near future. This version will minimally provide a stable selection of in-domain articles and gold-standard sentence segmentation. In joint work with the LinGO developers, we expect to adapt both the grammar (extend or improve linguistic analyses) and parsing technology in the light of the WeScience

```
tmp1=/tmp/.wescience.${USER}.$$.1
tmp2=/tmp/.wescience.${USER}.$$.2

cd ${LOGONROOT}/uio/wescience/bin;

Python ccp.py -i ${LOGONROOT}/uio/wescience/raw/ -o ${tmp1} -n;
${LOGONROOT}/cis/bin/linux.x86.32/tokenizer -L en-u8 -S ${tmp1} > ${tmp2};
Python cccp.py -i ${tmp2} -l 1000 -o ${LOGONROOT}/uio/wescience/txt;

/bin/rm -f ${tmp1} ${tmp2} > /dev/null;
```

Figure 8: The `Corpus Clean` pipeline in use.

experience.

References

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, 19:313 – 330, 1993.

Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575 – 596, 2004.

CJ Rupp, Ann Copestake, Simone Teufel, and Ben Waldron. Flexible interfaces in the application of language technology to an eScience corpus. In *Proceedings of the UK eScience Programme All Hands Meeting*, Nottingham, UK, 2007.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and J. Tsujii. Syntax annotation for the GENIA corpus. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 222 – 227, Jeju, Korea, 2005.

Gisle Ytrestol, Dan Flickinger, and Stephan Oepen. Extracting and Annotating Wikipedia Sub-Domains — Towards a New eScience Community Resource. In *Proceedings of The Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7 2009)*, Groningen, The Netherlands, 2009.

A List of Articles

A.1 The Articles in the WeScience Corpus

001.Algorithm
002.Ambiguity
003.Artificial intelligence
004.Artificial Linguistic Internet Computer Entity
005.Artificial neural network
006.Association for Computational Linguistics
007.Babel Fish (website)
008.Bioinformatics
009.BLEU
010.Business intelligence
011.Chatterbot
012.Computational linguistics
013.Computer program
014.Computer science
015.Corpus linguistics
016.Cross-platform
017.Data
018.Data analysis
019.Database
020.Cluster analysis
021.Data mining
022.Data set
023.ELIZA
024.English language
025.Esperanto
026.Formal grammar
027.Free software
028.Freeware
029.French language
030.German language
031.GNU General Public License
032.Google
033.Google Translate
034.Grammar
035.Hidden Markov model
036.HTML
037.IBM
038.Information
039.Information extraction
040.Information retrieval

041.Information theory
042.Italian language
043.Japanese language
044.Java (programming language)
045.Language
046.Language model
047.Latent semantic analysis
048.Linguistics
049.Linux
050.List of chatterbots
051.Loebner prize
052.Machine learning
053.Machine translation
054.Metadata
055.Microsoft Windows
056.Morphology (linguistics)
057.Named entity recognition
058.Natural language
059.Natural language processing
060.Neural network
061.N-gram
062.Noun
063.Ontology (information science)
064.OpenOffice.org
065.Parsing
066.Lexical category
067.Part-of-speech tagging
068.Pattern recognition
069.Phrase
070.Portuguese language
071.Predictive analytics
072.RapidMiner
073.Russian language
074.Web search engine
075.Semantics
076.Sentence (linguistics)
077.Computer software
078.Spanish language
079.Speech recognition
080.Speech synthesis
081.Statistical classification
082.Statistical machine translation
083.Statistics
084.Syntax

085.SYSTRAN
086.Text analytics
087.Text corpus
088.Text mining
089.Translation
090.Translation memory
091.Turing test
092.United States
093.Verb
094.Web application
095.Word
096.WordNet
097.WordPerfect
098.Word sense disambiguation
099.XHTML
100.XML

B Article Selection

B.1 Seed Articles

The 355 articles from Computational Linguistics or any of the subcategories:

http://127.0.0.1:8000/article/Able%20Danger
http://127.0.0.1:8000/article/Accuracy%20paradox
http://127.0.0.1:8000/article/Acoustic%20Model
http://127.0.0.1:8000/article/Additive%20smoothing
http://127.0.0.1:8000/article/Affinity%20analysis
http://127.0.0.1:8000/article/AFNLP
http://127.0.0.1:8000/article/Albert%20One
http://127.0.0.1:8000/article/Alicebot
http://127.0.0.1:8000/article/ALPAC
http://127.0.0.1:8000/article/Amarna%20letters
http://127.0.0.1:8000/article/American%20National%20Corpus
http://127.0.0.1:8000/article/Analogical%20modeling
http://127.0.0.1:8000/article/Anomaly%20detection
http://127.0.0.1:8000/article/Ants%20sleeping%20model
http://127.0.0.1:8000/article/AOLbyPhone
http://127.0.0.1:8000/article/Apatar
http://127.0.0.1:8000/article/Apertium
http://127.0.0.1:8000/article/Apriori%20algorithm
http://127.0.0.1:8000/article/Articulatory%20phonology
http://127.0.0.1:8000/article/Artificial%20grammar%20learning
http://127.0.0.1:8000/article/Artificial%20Linguistic%20Internet%
20Computer%20Entity
http://127.0.0.1:8000/article/Asia%20Online
http://127.0.0.1:8000/article/Association%20for%20Computational%
20Linguistics
http://127.0.0.1:8000/article/Association%20rule%20learning
http://127.0.0.1:8000/article/Audio%20mining
http://127.0.0.1:8000/article/Audio-visual%20speech%20recognition
http://127.0.0.1:8000/article/Augmented%20transition%20network
http://127.0.0.1:8000/article/Automated%20Lip%20Reading
http://127.0.0.1:8000/article/Automated%20Targeting%20System
http://127.0.0.1:8000/article/Automatic%20distillation%20of%20structure
http://127.0.0.1:8000/article/Automatic%20summarization
http://127.0.0.1:8000/article/Babel%20Fish%20(website)
http://127.0.0.1:8000/article/Babylon%20(program)
http://127.0.0.1:8000/article/Bag%20of%20words%20model
http://127.0.0.1:8000/article/Bank%20of%20English
http://127.0.0.1:8000/article/BaseN

http://127.0.0.1:8000/article/Benford\'s%20law
http://127.0.0.1:8000/article/Biclustering
http://127.0.0.1:8000/article/Bigram
http://127.0.0.1:8000/article/Bijankhan%20Corpus
http://127.0.0.1:8000/article/Biomedical%20text%20mining
http://127.0.0.1:8000/article/Bitext%20word%20alignment
http://127.0.0.1:8000/article/BLEU
http://127.0.0.1:8000/article/BowLingual
http://127.0.0.1:8000/article/Bradford\'s%20law
http://127.0.0.1:8000/article/Brill%20tagger
http://127.0.0.1:8000/article/British%20National%20Corpus
http://127.0.0.1:8000/article/Brown%20Corpus
http://127.0.0.1:8000/article/Business%20analytics
http://127.0.0.1:8000/article/Calais%20(Reuters%20Product)
http://127.0.0.1:8000/article/Center%20embedding
http://127.0.0.1:8000/article/Chart%20parser
http://127.0.0.1:8000/article/ChaSen
http://127.0.0.1:8000/article/CHAT%20and%20TIPS
http://127.0.0.1:8000/article/Chatterbot
http://127.0.0.1:8000/article/Chinese%20speech%20synthesis
http://127.0.0.1:8000/article/Chris%20Westphal
http://127.0.0.1:8000/article/ClearForest
http://127.0.0.1:8000/article/Cluster%20analysis
http://127.0.0.1:8000/article/CMU%20Pronouncing%20Dictionary
http://127.0.0.1:8000/article/Coh-Metrix
http://127.0.0.1:8000/article/Collocation
http://127.0.0.1:8000/article/Companions%20Project
http://127.0.0.1:8000/article/Comparison%20of%20machine%20translation%
20applications
http://127.0.0.1:8000/article/Computational%20humor
http://127.0.0.1:8000/article/Computational%20lexicology
http://127.0.0.1:8000/article/Computational%20linguistics
http://127.0.0.1:8000/article/Computational%20Linguistics%20(journal)
http://127.0.0.1:8000/article/Computational%20semantics
http://127.0.0.1:8000/article/Computer-assisted%20translation
http://127.0.0.1:8000/article/Concept%20drift
http://127.0.0.1:8000/article/Concept%20mining
http://127.0.0.1:8000/article/Concordancer
http://127.0.0.1:8000/article/Controlled%20natural%20language
http://127.0.0.1:8000/article/Conversational%20agent
http://127.0.0.1:8000/article/Co-occurrence%20networks
http://127.0.0.1:8000/article/Corpora%20(journal)
http://127.0.0.1:8000/article/Corpus%20linguistics
http://127.0.0.1:8000/article/Corpus%20Linguistics%20and%20Linguistic%

20Theory%20(journal)
http://127.0.0.1:8000/article/Corpus%20of%20Contemporary%20American%
20English
http://127.0.0.1:8000/article/Croatian%20National%20Corpus
http://127.0.0.1:8000/article/Cross%20Industry%20Standard%20Process%
20for%20Data%20Mining
http://127.0.0.1:8000/article/Cross-language%20information%20retrieval
http://127.0.0.1:8000/article/CrushConnect
http://127.0.0.1:8000/article/CSL%20(speech%20analysis)
http://127.0.0.1:8000/article/Cypher%20transcoder
http://127.0.0.1:8000/article/Data%20dredging
http://127.0.0.1:8000/article/Data%20fusion
http://127.0.0.1:8000/article/Data%20mining
http://127.0.0.1:8000/article/Data%20mining%20agent
http://127.0.0.1:8000/article/Data-snooping%20bias
http://127.0.0.1:8000/article/Data%20stream%20mining
http://127.0.0.1:8000/article/Data%20visualization
http://127.0.0.1:8000/article/Decision%20tree%20learning
http://127.0.0.1:8000/article/Deep%20Web%20Technologies
http://127.0.0.1:8000/article/DialogOS
http://127.0.0.1:8000/article/Dictionary-based%20machine%20translation
http://127.0.0.1:8000/article/Distributed%20Language%20Translation
http://127.0.0.1:8000/article/Distributional%20hypothesis
http://127.0.0.1:8000/article/Document%20classification
http://127.0.0.1:8000/article/Document-term%20matrix
http://127.0.0.1:8000/article/Dr.%20Sbaitso
http://127.0.0.1:8000/article/Dynamic%20itemset%20counting
http://127.0.0.1:8000/article/Early%20stopping
http://127.0.0.1:8000/article/Ebla%20tablets
http://127.0.0.1:8000/article/Elbot
http://127.0.0.1:8000/article/ELIZA
http://127.0.0.1:8000/article/Espionage
http://127.0.0.1:8000/article/ETBLAST
http://127.0.0.1:8000/article/Eurotra
http://127.0.0.1:8000/article/EuroWordNet
http://127.0.0.1:8000/article/Evaluation%20of%20machine%20translation
http://127.0.0.1:8000/article/Example-based%20machine%20translation
http://127.0.0.1:8000/article/EXtended%20WordNet
http://127.0.0.1:8000/article/Factored%20language%20model
http://127.0.0.1:8000/article/FatiGO
http://127.0.0.1:8000/article/FLAME%20clustering
http://127.0.0.1:8000/article/Foreign%20language%20reading%20aid
http://127.0.0.1:8000/article/Foreign%20language%20writing%20aid
http://127.0.0.1:8000/article/Formal%20concept%20analysis

http://127.0.0.1:8000/article/FrameNet
http://127.0.0.1:8000/article/Fred%20Chatterbot
http://127.0.0.1:8000/article/Frederick%20Jelinek
http://127.0.0.1:8000/article/Frequency%20list
http://127.0.0.1:8000/article/GEMET
http://127.0.0.1:8000/article/General%20Architecture%20for%20Text%
20Engineering
http://127.0.0.1:8000/article/GeneRIF
http://127.0.0.1:8000/article/Genetic%20fuzzy%20systems
http://127.0.0.1:8000/article/Georgetown-IBM%20experiment
http://127.0.0.1:8000/article/Google%20Translate
http://127.0.0.1:8000/article/Grammar%20induction
http://127.0.0.1:8000/article/GramTrans
http://127.0.0.1:8000/article/Group%20method%20of%20data%20handling
http://127.0.0.1:8000/article/GSP%20Algorithm
http://127.0.0.1:8000/article/Hamshahri%20Corpus
http://127.0.0.1:8000/article/HAREM
http://127.0.0.1:8000/article/Heaps\'%20law
http://127.0.0.1:8000/article/History%20of%20machine%20translation
http://127.0.0.1:8000/article/Inference%20attack
http://127.0.0.1:8000/article/Information%20extraction
http://127.0.0.1:8000/article/Information%20Harvesting
http://127.0.0.1:8000/article/Institute%20of%20Analytics%20Professionals%
20of%20Australia
http://127.0.0.1:8000/article/Intelligent%20character%20recognition
http://127.0.0.1:8000/article/Interlingual%20machine%20translation
http://127.0.0.1:8000/article/International%20Corpus%20of%20English
http://127.0.0.1:8000/article/International%20Journal%20of%20Corpus%
20Linguistics
http://127.0.0.1:8000/article/IntraText
http://127.0.0.1:8000/article/Jabberwacky
http://127.0.0.1:8000/article/Java%20Machine%20Learning%20Library
http://127.0.0.1:8000/article/John%20Lennon%20Artificial%20Intelligence%
20Project
http://127.0.0.1:8000/article/Julius%20(software)
http://127.0.0.1:8000/article/Kalle%20Kotipsykiatri
http://127.0.0.1:8000/article/Katz\'s%20back-off%20model
http://127.0.0.1:8000/article/Kdd%20Ontology
http://127.0.0.1:8000/article/Keyword%20(linguistics)
http://127.0.0.1:8000/article/K-optimal%20pattern%20discovery
http://127.0.0.1:8000/article/Kultepe%20texts
http://127.0.0.1:8000/article/KXEN%20Inc.
http://127.0.0.1:8000/article/Language%20and%20Computers
http://127.0.0.1:8000/article/Language%20engineering

http://127.0.0.1:8000/article/Language%20identification
http://127.0.0.1:8000/article/Language%20model
http://127.0.0.1:8000/article/Language%20recognition
http://127.0.0.1:8000/article/Language%20Technologies%20Institute
http://127.0.0.1:8000/article/Languageware
http://127.0.0.1:8000/article/Language%20Weaver
http://127.0.0.1:8000/article/Larry%20E.%20Smith
http://127.0.0.1:8000/article/Latent%20Dirichlet%20allocation
http://127.0.0.1:8000/article/Latent%20semantic%20analysis
http://127.0.0.1:8000/article/Latent%20semantic%20mapping
http://127.0.0.1:8000/article/Lemmatisation
http://127.0.0.1:8000/article/Lesk%20algorithm
http://127.0.0.1:8000/article/Levenshtein%20automaton
http://127.0.0.1:8000/article/Lexical%20Markup%20Framework
http://127.0.0.1:8000/article/Lift%20(data%20mining)
http://127.0.0.1:8000/article/Linguaphile
http://127.0.0.1:8000/article/LinguaStream
http://127.0.0.1:8000/article/Linguatec
http://127.0.0.1:8000/article/Linguistic%20Data%20Consortium
http://127.0.0.1:8000/article/Linguistic%20Issues%20in%20Language%
20Technology
http://127.0.0.1:8000/article/Linguistic%20Knowledge%20Builder
http://127.0.0.1:8000/article/Linguistics%20Research%20Center%20at%
20UT%20Austin
http://127.0.0.1:8000/article/Link%20grammar
http://127.0.0.1:8000/article/Link%20Grammar%20Parser
http://127.0.0.1:8000/article/List%20of%20chatterbots
http://127.0.0.1:8000/article/List%20of%20research%20laboratories%
20for%20machine%20translation
http://127.0.0.1:8000/article/Logic%20form
http://127.0.0.1:8000/article/LOLITA
http://127.0.0.1:8000/article/LREC
http://127.0.0.1:8000/article/Lbke%20English
http://127.0.0.1:8000/article/Machine-readable%20dictionary
http://127.0.0.1:8000/article/Machine%20translation
http://127.0.0.1:8000/article/Machine%20translation%20software%
20usability
http://127.0.0.1:8000/article/Mari%20Tablets
http://127.0.0.1:8000/article/MBROLA
http://127.0.0.1:8000/article/MDic
http://127.0.0.1:8000/article/MegaHAL
http://127.0.0.1:8000/article/Message%20Understanding%20Conference
http://127.0.0.1:8000/article/METAL%20MT
http://127.0.0.1:8000/article/METEOR

http://127.0.0.1:8000/article/METEO%20System
http://127.0.0.1:8000/article/Microsoft%20Anna
http://127.0.0.1:8000/article/Modular%20Audio%20Recognition%20Framework
http://127.0.0.1:8000/article/Molecule%20mining
http://127.0.0.1:8000/article/Morphological%20dictionary
http://127.0.0.1:8000/article/Morphological%20pattern
http://127.0.0.1:8000/article/Moses%20(machine%20translation)
http://127.0.0.1:8000/article/Multi-document%20summarization
http://127.0.0.1:8000/article/Multilingual%20notation
http://127.0.0.1:8000/article/Named%20entity%20recognition
http://127.0.0.1:8000/article/National%20Centre%20for%20Text%20Mining
http://127.0.0.1:8000/article/Natural%20computation
http://127.0.0.1:8000/article/Natural%20language
http://127.0.0.1:8000/article/Natural%20language%20generation
http://127.0.0.1:8000/article/Natural%20language%20processing
http://127.0.0.1:8000/article/Nearest%20neighbor%20search
http://127.0.0.1:8000/article/Neo-Assyrian%20Text%20Corpus%20Project
http://127.0.0.1:8000/article/Neural%20network
http://127.0.0.1:8000/article/N-gram
http://127.0.0.1:8000/article/NIST%20(metric)
http://127.0.0.1:8000/article/Noisy%20channel%20model
http://127.0.0.1:8000/article/Noisy%20text%20analytics
http://127.0.0.1:8000/article/Nora%20(technology)
http://127.0.0.1:8000/article/North%20American%20Association%20for%
20Computational%20Linguistics
http://127.0.0.1:8000/article/Ontology%20learning
http://127.0.0.1:8000/article/Open%20domain
http://127.0.0.1:8000/article/OpenLogos
http://127.0.0.1:8000/article/Open%20Source%20Intelligence
http://127.0.0.1:8000/article/Open%20Translation%20Engine
http://127.0.0.1:8000/article/Optical%20character%20recognition
http://127.0.0.1:8000/article/Optimal%20matching
http://127.0.0.1:8000/article/Overfitting
http://127.0.0.1:8000/article/Oxford%20English%20Corpus
http://127.0.0.1:8000/article/Paco%20Nathan
http://127.0.0.1:8000/article/PARRY
http://127.0.0.1:8000/article/Part-of-speech%20tagging
http://127.0.0.1:8000/article/PATR-II
http://127.0.0.1:8000/article/Pattern%20mining
http://127.0.0.1:8000/article/Phrasal%20template
http://127.0.0.1:8000/article/Phrase%20chunking
http://127.0.0.1:8000/article/Phraselator
http://127.0.0.1:8000/article/Praat
http://127.0.0.1:8000/article/Principal%20components%20analysis

http://127.0.0.1:8000/article/Probabilistic%20latent%20semantic%
20analysis
http://127.0.0.1:8000/article/Production%20(computer%20science)
http://127.0.0.1:8000/article/Promt
http://127.0.0.1:8000/article/PropBank
http://127.0.0.1:8000/article/Prospero%20Business%20Suite
http://127.0.0.1:8000/article/Q-systems
http://127.0.0.1:8000/article/Quack.com
http://127.0.0.1:8000/article/Question%20answering
http://127.0.0.1:8000/article/Racter
http://127.0.0.1:8000/article/RapidMiner
http://127.0.0.1:8000/article/Receiver%20operating%20characteristic
http://127.0.0.1:8000/article/Recursive%20transition%20network
http://127.0.0.1:8000/article/Relationship%20extraction
http://127.0.0.1:8000/article/Rewrite%20rule
http://127.0.0.1:8000/article/Robby%20Garner
http://127.0.0.1:8000/article/Round-trip%20translation
http://127.0.0.1:8000/article/Russian%20National%20Corpus
http://127.0.0.1:8000/article/SABLE
http://127.0.0.1:8000/article/SALERO
http://127.0.0.1:8000/article/Scottish%20Corpus%20of%20Texts%20and%
20Speech
http://127.0.0.1:8000/article/Scriptella
http://127.0.0.1:8000/article/Semantic%20analytics
http://127.0.0.1:8000/article/Semantic%20neural%20network
http://127.0.0.1:8000/article/Semantic%20prosody
http://127.0.0.1:8000/article/Semantic%20relatedness
http://127.0.0.1:8000/article/Semantic%20similarity
http://127.0.0.1:8000/article/Sentence%20boundary%20disambiguation
http://127.0.0.1:8000/article/Sentence%20extraction
http://127.0.0.1:8000/article/Shallow%20parsing
http://127.0.0.1:8000/article/Sinewave%20synthesis
http://127.0.0.1:8000/article/Sinkov%20statistic
http://127.0.0.1:8000/article/Sliding%20window%20based%20part-of-speech%
20tagging
http://127.0.0.1:8000/article/SmarterChild
http://127.0.0.1:8000/article/Snack%20audio%20library
http://127.0.0.1:8000/article/Snikers
http://127.0.0.1:8000/article/Software%20mining
http://127.0.0.1:8000/article/Soundex
http://127.0.0.1:8000/article/Speech%20corpus
http://127.0.0.1:8000/article/Speech%20recognition
http://127.0.0.1:8000/article/Speech%20segmentation
http://127.0.0.1:8000/article/Speech%20synthesis

http://127.0.0.1:8000/article/Spleak
http://127.0.0.1:8000/article/SPL%20notation
http://127.0.0.1:8000/article/Spoken%20dialog%20system
http://127.0.0.1:8000/article/StarDict
http://127.0.0.1:8000/article/Statistical%20machine%20translation
http://127.0.0.1:8000/article/Statistical%20parsing
http://127.0.0.1:8000/article/Statistical%20semantics
http://127.0.0.1:8000/article/Stefan%20Th.%20Gries
http://127.0.0.1:8000/article/Stemming
http://127.0.0.1:8000/article/Stochastic%20context-free%20grammar
http://127.0.0.1:8000/article/Stochastic%20grammar
http://127.0.0.1:8000/article/Structure%20mining
http://127.0.0.1:8000/article/Studies%20in%20NLP
http://127.0.0.1:8000/article/Subvocal%20recognition
http://127.0.0.1:8000/article/Sukhotins%20Algorithm
http://127.0.0.1:8000/article/Survey%20of%20English%20Usage
http://127.0.0.1:8000/article/Syllabotactics
http://127.0.0.1:8000/article/Synthetix
http://127.0.0.1:8000/article/SYSTRAN
http://127.0.0.1:8000/article/Talkman
http://127.0.0.1:8000/article/Targumatik
http://127.0.0.1:8000/article/TAUM%20system
http://127.0.0.1:8000/article/Technolangue/Easy
http://127.0.0.1:8000/article/Teragram%20Corporation
http://127.0.0.1:8000/article/Terminology%20extraction
http://127.0.0.1:8000/article/Text%20analytics
http://127.0.0.1:8000/article/Text%20corpus
http://127.0.0.1:8000/article/Text%20mining
http://127.0.0.1:8000/article/Text%20Retrieval%20Conference
http://127.0.0.1:8000/article/Text%20segmentation
http://127.0.0.1:8000/article/Text%20simplification
http://127.0.0.1:8000/article/Tfidf
http://127.0.0.1:8000/article/Thesaurus%20Linguae%20Graecae
http://127.0.0.1:8000/article/TIMIT
http://127.0.0.1:8000/article/Traduwiki
http://127.0.0.1:8000/article/Transcriber
http://127.0.0.1:8000/article/Transderivational%20search
http://127.0.0.1:8000/article/Transfer-based%20machine%20translation
http://127.0.0.1:8000/article/Treatment%20learner
http://127.0.0.1:8000/article/Treebank
http://127.0.0.1:8000/article/Trigram
http://127.0.0.1:8000/article/Trigram%20tagger
http://127.0.0.1:8000/article/Ultra%20Hal%20Assistant
http://127.0.0.1:8000/article/United%20Nations%20Multilingual%20Terminology%

```
20Database
http://127.0.0.1:8000/article/Universal%20Networking%20Language
http://127.0.0.1:8000/article/User:Stevenbird/List%20of%20NLP%20Courses
http://127.0.0.1:8000/article/Variable%20rules%20analysis
http://127.0.0.1:8000/article/VerbAce
http://127.0.0.1:8000/article/Verbmobil
http://127.0.0.1:8000/article/VerbNet
http://127.0.0.1:8000/article/Verbot
http://127.0.0.1:8000/article/Virtual%20Woman
http://127.0.0.1:8000/article/Voice%20activity%20detection
http://127.0.0.1:8000/article/Voice%20output%20communication%20aid
http://127.0.0.1:8000/article/VoxForge
http://127.0.0.1:8000/article/WaveSurfer
http://127.0.0.1:8000/article/Weather%20Data%20Mining
http://127.0.0.1:8000/article/Webclopedia
http://127.0.0.1:8000/article/Web%20mining
http://127.0.0.1:8000/article/Weidner%20Communications
http://127.0.0.1:8000/article/Weka%20(machine%20learning)
http://127.0.0.1:8000/article/Windows%20Live%20Translator
http://127.0.0.1:8000/article/Wordfast
http://127.0.0.1:8000/article/WordNet
http://127.0.0.1:8000/article/Word%20sense%20disambiguation
http://127.0.0.1:8000/article/Wrapper%20(data%20mining)
http://127.0.0.1:8000/article/Writer%20invariant
http://127.0.0.1:8000/article/W-shingling
http://127.0.0.1:8000/article/Zeta%20distribution
http://127.0.0.1:8000/article/ZipfMandelbrot%20law
http://127.0.0.1:8000/article/Zipf\'s%20law
```

B.2 Number of Links

The number of links in the seed articles (see section B.1), derived by the Python script `linkextractor.py`.

Usage: `python linkextractor.py seedArticles > seedArticles.links`

```
('/article/Natural_language_processing/',84)
('/article/Machine_translation/',73)
('/article/Data_mining/',53)
('/article/Computational_linguistics/',47)
('/article/Speech_recognition/',43)
('/article/English_language/',36)
('/article/Machine_learning/',35)
```

```
('/article/Artificial_intelligence/',34)
('/article/Text_corpus/',32)
('/article/Information_retrieval/',30)
('/article/Natural_language/',29)
('/article/XML/',28)
('/article/Linguistics/',25)
('/article/Corpus_linguistics/',25)
('/article/Proprietary/',24)
('/article/Chatterbot/',23)
('/article/HTML/',22)
('/article/Text_mining/',21)
('/article/Statistical_machine_translation/',21)
('/article/Database/',21)
('/article/Statistics/',20)
('/article/N-gram/',20)
('/article/Information_extraction/',20)
('/article/Syntax/',19)
('/article/Speech_synthesis/',19)
('/article/Part-of-speech_tagging/',18)
('/article/Java_%28programming_language%29/',18)
('/article/2005/',18)
('/article/Russian_language/',17)
('/article/WordNet/',16)
('/article/Translation_memory/',16)
('/article/Parsing/',16)
('/article/French_language/',16)
('/article/Translation/',15)
('/article/Grammar/',15)
('/article/Spanish_language/',14)
('/article/IBM/',14)
('/article/GNU_General_Public_License/',14)
('/article/Algorithm/',14)
('/article/Web_application/',13)
('/article/Semantics/',13)
('/article/SYSTRAN/',13)
('/article/Ontology_%28computer_science%29/',13)
('/article/Natural_Language_Processing/',13)
('/article/Loebner_Prize/',13)
('/article/Google/',13)
('/article/Artificial_neural_network/',13)
('/article/OpenOffice.org/',12)
('/article/Multiplatform/',12)
('/article/Latent_semantic_analysis/',12)
('/article/Language/',12)
```

```
('/article/German_language/',12)
('/article/ELIZA/',12)
('/article/Babel_Fish_%28website%29/',12)
('/article/Morphology_%28linguistics%29/',11)
('/article/Microsoft_Windows/',11)
('/article/Metadata/',11)
('/article/Word/',10)
('/article/Systran/',10)
('/article/Software/',10)
('/article/Sentence_%28linguistics%29/',10)
('/article/Portuguese_language/',10)
('/article/Pattern_recognition/',10)
('/article/Noun/',10)
('/article/Named_entity_recognition/',10)
('/article/Hidden_Markov_model/',10)
('/article/Google_Translate/',10)
('/article/Freeware/',10)
('/article/Free_software/',10)
('/article/Data_set/',10)
('/article/Data/',10)
('/article/Cross-platform/',10)
('/article/Word_sense_disambiguation/',9)
('/article/WordPerfect/',9)
('/article/United_States/',9)
('/article/Turing_test/',9)
('/article/Text-to-speech/',9)
('/article/Statistical_classification/',9)
('/article/Search_engine/',9)
('/article/Predictive_analytics/',9)
('/article/Phrase/',9)
('/article/Linux/',9)
('/article/Information_theory/',9)
('/article/Data_clustering/',9)
('/article/Computer_program/',9)
('/article/British_National_Corpus/',9)
('/article/Artificial_Linguistic_Internet_Computer_Entity/',9)
('/article/XHTML/',8)
('/article/Verb/',8)
('/article/Text_analytics/',8)
('/article/RapidMiner/',8)
('/article/Part_of_speech/',8)
('/article/Neural_network/',8)
('/article/Loebner_prize/',8)
('/article/List_of_Chatterbots/',8)
```

```
('/article/Language_model/',8)
('/article/Japanese_language/',8)
('/article/Italian_language/',8)
('/article/Information/',8)
('/article/Formal_grammar/',8)
('/article/Esperanto/',8)
('/article/Data_analysis/',8)
('/article/Corpus/',8)
('/article/Computer_science/',8)
('/article/Business_intelligence/',8)
('/article/Bioinformatics/',8)
('/article/BLEU/',8)
('/article/Association_for_Computational_Linguistics/',8)
('/article/Ambiguity/',8)
('/article/2006/',8)
```

B.3 Remaining articles in the corpus

The stored articles, using the Python script `makecorpus.py`.

Usage: `python makecorpus.py seedArticles.links`

001.Algorithm.source
002.Ambiguity.source
003.Artificial intelligence.source
004.Artificial Linguistic Internet Computer Entity.source
005.Artificial neural network.source
006.Association for Computational Linguistics.source
007.Babel Fish (website).source
008.Bioinformatics.source
009.BLEU.source
010.Business intelligence.source
011.Chatterbot.source
012.Computational linguistics.source
013.Computer program.source
014.Computer science.source
015.Corpus linguistics.source
016.Cross-platform.source
017.Data.source
018.Data analysis.source
019.Database.source
020.Cluster analysis.source
021.Data mining.source

022.Data set.source
023.ELIZA.source
024.English language.source
025.Esperanto.source
026.Formal grammar.source
027.Free software.source
028.Freeware.source
029.French language.source
030.German language.source
031.GNU General Public License.source
032.Google.source
033.Google Translate.source
034.Grammar.source
035.Hidden Markov model.source
036.HTML.source
037.IBM.source
038.Information.source
039.Information extraction.source
040.Information retrieval.source
041.Information theory.source
042.Italian language.source
043.Japanese language.source
044.Java (programming language).source
045.Language.source
046.Language model.source
047.Latent semantic analysis.source
048.Linguistics.source
049.Linux.source
050.List of chatterbots.source
051.Loebner prize.source
052.Machine learning.source
053.Machine translation.source
054.Metadata.source
055.Microsoft Windows.source
056.Morphology (linguistics).source
057.Named entity recognition.source
058.Natural language.source
059.Natural language processing.source
060.Neural network.source
061.N-gram.source
062.Noun.source
063.Ontology (information science).source
064.OpenOffice.org.source
065.Parsing.source

066.Lexical category.source
067.Part-of-speech tagging.source
068.Pattern recognition.source
069.Phrase.source
070.Portuguese language.source
071.Predictive analytics.source
072.RapidMiner.source
073.Russian language.source
074.Web search engine.source
075.Semantics.source
076.Sentence (linguistics).source
077.Computer software.source
078.Spanish language.source
079.Speech recognition.source
080.Speech synthesis.source
081.Statistical classification.source
082.Statistical machine translation.source
083.Statistics.source
084.Syntax.source
085.SYSTRAN.source
086.Text analytics.source
087.Text corpus.source
088.Text mining.source
089.Translation.source
090.Translation memory.source
091.Turing test.source
092.United States.source
093.Verb.source
094.Web application.source
095.Word.source
096.WordNet.source
097.WordPerfect.source
098.Word sense disambiguation.source
099.XHTML.source
100.XML.source

C Alternative Link Analysis Cycle

It seems the naming convention in Wikipedia Source Markup suffer from two major inconsistency issues: The usage of camel case ("Word Processor") rather than capitalizing only the first letter ("Word processor") and the use of whitespace/underscore between two words ("Artificial_intelligence" and "Artificial intelligence"). I have made an improved version of `link extractor` that compensates for this inconsistency in the naming convention of articles by counting the pages that are being redirected to. Had I used this improvised link analysis, the article selection would have differed to some extent, but only for the "least relevant" articles, e.g. the articles with the fewest references.

By using this new link analysis, I extracted the 112 most frequent article references from the seed articles, and compared against the original 110 articles used by the WeScience Corpus. Of the 110 articles that were used by WeScience, there were only 9 articles that were not among the 112 articles extracted by the improved link analyzer[7].

Of the 112 articles extracted by the new link extraction cycle, there were 17 articles with 9 references or more that were not among the original selection in WeScience. 10 of these articles had the minimum of 9 references, and it should be noted that not all 110 articles in the original selection are unique, i.e. some of the links refer to the same article (like *Cross-platform* and *Multiplatform*), and if we actually were to use the improved link analyzer to extract article candidates, we could not have included all articles with 9 references, but would probably have set the threshold to 10.

We did however decide to stick to the original selection, instead of scraping the initial work and start over with a new selection. When dealing with 18 GB of user-generated data, compensating for inconsistencies in the dataset seems like a never-ending task, and we have therefore decided that our original selection is acceptable, and there would be little to gain from starting over again. To truly compensate for the all inconsistencies in the dataset does not seem feasible. Even in the alternative version of counting redirects, we detected inconsistencies with respect to the naming convention, e.g. will the the entry "AI" redirect to the page "Artificial intelligence", whereas the page "Artificial Intelligence" will redirect to the page "Artificial_intelligence" (with underscore).

C.1 How It Was Carried Out

By using an alternative version of `link extractor`, I retrieved a selection which differ slightly from the original WeScience selection listed in Section B.2.

[7] These articles were: *Data analysis*, *Esperanto*, *Information*, *Italian language*, *Japanese_language*, *RapidMiner*, *Word* and *XHTML*

When using this new script, we first have to generate a Python dictionary where the redirects if the original Wikipedia dump is stored. This is done in two stages. First we retrieve all the redirects by using a grep command:

```
egrep -B 20 -i '#redirect' enwiki-latest-pages-articles.xml |
egrep -i '<title>|#redirect' > redirect.txt
```

`redirect.txt` now contains all redirects in the Wikipedia dump. We parse `redirect.txt` with `linkDict.py` to retrieve a Python dictionary that maps all the article names to the redirect link, e.g. so that both "Part-of-speech" and "Part of Speech" now will have the same hash key (namely "Lexical category").

C.2 Article Frequency in Alternative Link Extraction

Usage:[8]

1. `cat redirect.txt | python linkDict.py`
2. `python linkextractor.newapproach.py seedArticles > seedArticles.links`

('natural language processing', 97)
('machine translation', 83)
('data mining', 53)
('computational linguistics', 52)
('speech recognition', 51)
('english language', 39)
('machine learning', 37)
('artificial_intelligence', 37)
('2005', 36)
('speech synthesis', 35)
('text corpus', 34)
('natural language', 33)
('information retrieval', 31)
('statistics', 30)
('linguistics', 30)
('xml', 29)
('cross-platform', 28)
('chatterbot', 26)
('proprietary', 25)
('n-gram', 25)
('database', 25)
('systran', 24)

[8] `linkDict.py` and `linkextractor.newapproach.py` can be made available upon request.

('syntax', 24)
('corpus linguistics', 24)
('translation', 23)
('parsing', 23)
('gnu general public license', 23)
('text mining', 22)
('java (programming language)', 22)
('html', 22)
('statistical machine translation', 21)
('loebner prize', 20)
('information extraction', 20)
('wordnet', 19)
('semantics', 19)
('part-of-speech tagging', 19)
('2006', 19)
('russian language', 18)
('grammar', 18)
('translation memory', 17)
('computer software', 17)
('microsoft windows', 16)
('french language', 16)
('algorithm', 16)
('spanish language', 15)
('latent semantic analysis', 15)
('eigenvalue, eigenvector and eigenspace', 15)
('united states', 14)
('lexical category', 14)
('ibm', 14)
('google', 14)
('eliza', 14)
('bleu', 14)
('artificial neural network', 14)
('web search engine', 13)
('web application', 13)
('synonym', 13)
('named entity recognition', 13)
('linux', 13)
('language', 13)
('corpus', 13)
('babel fish (website)', 13)
('openoffice.org', 12)
('hidden markov model', 12)
('german language', 12)
('word sense disambiguation', 11)

('turing test', 11)
('morphology (linguistics)', 11)
('metadata', 11)
('list of chatterbots', 11)
('freeware', 11)
('free software', 11)
('cluster analysis', 11)
("zipf's law", 10)
('yahoo!', 10)
('wordperfect', 10)
('word processor', 10)
('word (disambiguation)', 10)
('statistical classification', 10)
('speech application programming interface', 10)
('sentence (linguistics)', 10)
('predictive analytics', 10)
('portuguese language', 10)
('pattern_recognition', 10)
('noun', 10)
('language model', 10)
('google translate', 10)
('data set', 10)
('data', 10)
('comparison of machine translation applications', 10)
('business intelligence', 10)
('verb', 9)
('phrase', 9)
('optical_character_recognition', 9)
('open source', 9)
('ontology (information science)', 9)
('microsoft word', 9)
('medline', 9)
('list of google products', 9)
('information theory', 9)
('finite state machine', 9)
('document classification', 9)
('computer-assisted translation', 9)
('computer science', 9)
('computer program', 9)
('computer', 9)
('carnegie mellon university', 9)
('british national corpus', 9)
('bioinformatics', 9)
('association for computational linguistics', 9)

('artificial linguistic internet computer entity', 9)
('ambiguity', 9)

C.3 New Articles in the Alternative Cycle

The top 112 articles retrived by the alternative link extractor. The articles with a "–" prefix is not a part of the original WeScience selection.

+('2005', 36)
+('2006', 19)
+('algorithm', 16)
+('ambiguity', 9)
+('artificial_intelligence', 37)
+('artificial linguistic internet computer entity', 9)
+('artificial neural network', 14)
+('association for computational linguistics', 9)
+('babel fish (website)', 13)
+('bioinformatics', 9)
+('bleu', 14)
+('british national corpus', 9)
+('business intelligence', 10)
–('carnegie mellon university', 9)
+('chatterbot', 26)
+('cluster analysis', 11)
–('comparison of machine translation applications', 10)
+('computational linguistics', 52)
–('computer', 9)
–('computer-assisted translation', 9)
+('computer program', 9)
+('computer science', 9)
+('computer software', 17)
+('corpus', 13)
+('corpus linguistics', 24)
+('cross-platform', 28)
+('data', 10)
+('database', 25)
+('data mining', 53)
+('data set', 10)
–('document classification', 9)
–('eigenvalue, eigenvector and eigenspace', 15)
+('eliza', 14)
+('english language', 39)
–('finite state machine', 9)

+('free software', 11)
+('freeware', 11)
+('french language', 16)
+('german language', 12)
+('gnu general public license', 23)
+('google', 14)
+('google translate', 10)
+('grammar', 18)
+('hidden markov model', 12)
+('html', 22)
+('ibm', 14)
+('information extraction', 20)
+('information retrieval', 31)
+('information theory', 9)
+('java (programming language)', 22)
+('language', 13)
+('language model', 10)
+('latent semantic analysis', 15)
+('lexical category', 14)
+('linguistics', 30)
+('linux', 13)
+('list of chatterbots', 11)
–('list of google products', 9)
+('loebner prize', 20)
+('machine learning', 37)
+('machine translation', 83)
–('medline', 9)
+('metadata', 11)
+('microsoft windows', 16)
–('microsoft word', 9)
+('morphology (linguistics)', 11)
+('named entity recognition', 13)
+('natural language', 33)
+('natural language processing', 97)
+('n-gram', 25)
+('noun', 10)
+('ontology (information science)', 9)
+('openoffice.org', 12)
–('open source', 9)
–('optical_character_recognition', 9)
+('parsing', 23)
+('part-of-speech tagging', 19)
+('pattern_recognition', 10)
+('phrase', 9)

+('portuguese language', 10)
+('predictive analytics', 10)
+('proprietary', 25)
+('russian language', 18)
+('semantics', 19)
+('sentence (linguistics)', 10)
+('spanish language', 15)
–('speech application programming interface', 10)
(+'speech recognition', 51)
(+'speech synthesis', 35)
(+'statistical classification', 10)
+('statistical machine translation', 21)
+('statistics', 30)
–('synonym', 13)
+('syntax', 24)
+('systran', 24)
+('text corpus', 34)
+('text mining', 22)
+('translation', 23)
+('translation memory', 17)
+('turing test', 11)
+('united states', 14)
+('verb', 9)
+('web application', 13)
+('web search engine', 13)
+('word (disambiguation)', 10)
+('wordnet', 19)
+('wordperfect', 10)
–('word processor', 10)
+('word sense disambiguation', 11)
+('xml', 29)
–('yahoo!', 10)
–("zipf's law", 10)

C.4 Articles in the Original WeScience Selection

Articles with a ++ prefix did not occur in the alternative article selection cycle.

('2005/', 18)
('2006/', 8)
('Algorithm/', 14)
('Ambiguity/', 8)
('Artificial_intelligence/', 34)
('Artificial_Linguistic_Internet_Computer_Entity/', 9)
('Artificial_neural_network/', 13)
('Association_for_Computational_Linguistics/', 8)
('Babel_Fish_*('Bioinformatics/', 8)
('BLEU/', 8)
('British_National_Corpus/', 9)
('Business_intelligence/', 8)
('Chatterbot/', 23)
('Computational_linguistics/', 47)
('Computer_program/', 9)
('Computer_science/', 8)
('Corpus/', 8)
('Corpus_linguistics/', 25)
('Cross-platform/', 10)
('Data/', 10)
++('Data_analysis/', 8)
('Database/', 21)
(cluster analysis)('Data_clustering/', 9)
('Data_mining/', 53)
('Data_set/', 10)
('ELIZA/', 12)
('English_language/', 36)
++('Esperanto/', 8)
++('Formal_grammar/', 8)
('Free_software/', 10)
('Freeware/', 10)
('French_language/', 16)
('German_language/', 12)
('GNU_General_Public_License/', 14)
('Google/', 13)
('Google_Translate/', 10)
('Grammar/', 15)
('Hidden_Markov_model/', 10)
('HTML/', 22)
('IBM/', 14)

++('Information/', 8)
('Information_extraction/', 20)
('Information_retrieval/', 30)
('Information_theory/', 9)
++('Italian_language/', 8)
++('Japanese_language/', 8)
('Java_%28programming_language%29/', 18)
('Language/', 12)
('Language_model/', 8)
('Latent_semantic_analysis/', 12)
('Linguistics/', 25)
('Linux/', 9)
('List_of_Chatterbots/', 8)
('Loebner_Prize/', 13)
('Loebner_prize/', 8)
('Machine_learning/', 35)
('Machine_translation/', 73)
('Metadata/', 11)
('Microsoft_Windows/', 11)
('Morphology_*(cross-platform)('Multiplatform/', 12)
('Named_entity_recognition/', 10)
('Natural_language/', 29)
('Natural_Language_Processing/', 13)
('Natural_language_processing/', 84)
(artificial)('Neural_network/', 8)
('N-gram/', 20)
('Noun/', 10)
('Ontology_%28computer_science%29/', 13)
('OpenOffice.org/', 12)
('Parsing/', 16)
('Part_of_speech/', 8)
('Part-of-speech_tagging/', 18)
('Pattern_recognition/', 10)
('Phrase/', 9)
('Portuguese_language/', 10)
('Predictive_analytics/', 9)
('Proprietary/', 24)

++('RapidMiner/', 8)
('Russian_language/', 17)
(web)('Search_engine/', 9)
('Semantics/', 13)
('Sentence_%28linguistics%29/', 10)
('Software/', 10)
('Spanish_language/', 14)
('Speech_recognition/', 43)
('Speech_synthesis/', 19)
('Statistical_classification/', 9)
('Statistical_machine_translation/', 21)
('Statistics/', 20)
('Syntax/', 19)
('Systran/', 10)
('SYSTRAN/', 13)
(text mining)('Text_analytics/', 8)
('Text_corpus/', 32)
('Text_mining/', 21)
(speech synt)('Text-to-speech/', 9)
('Translation/', 15)
('Translation_memory/', 16)
('Turing_test/', 9)
('United_States/', 9)
('Verb/', 8)
('Web_application/', 13)
++('Word/', 10)
('WordNet/', 16)
('WordPerfect/', 9)
('Word_sense_disambiguation/', 9)
++('XHTML/', 8)
('XML/', 28)