Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Martin Majliš

Large Multilingual Corpus

Institute of Formal and Applied Linguistics

Supervisor:	doc. Ing. Zdeněk Žabokrtský, Ph.D.
Study programme:	Informatics
Specialization:	Mathematical Linguistics

Prague 2011

I would like to thank my supervisor doc. Ing. Zdeněk Žabokrtský, Ph.D. for his advice and my parents for their support.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, August 5, 2011 ....

Název práce: Velký mnohojazyčný korpus

Autor: Martin Majliš

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: doc. Ing. Zdeněk Žabokrtský Ph.D.

Abstrakt: V této diplomové práci je popsán webový korpus W2C. Tento korpus obsahuje 97 jazyku a pro každý z nich alespoň 10 milionů slov. Celková velikost je 10,5 miliardy slov. Aby bylo možné takovýto korpus vytvořit, bylo nutné vyřešit celou řadu dílčích problémů. Na začátku musel být sestaven korpus z Wikipedie se 122 jazyky, na kterém byl natrénován rozpoznávač jazyků. Pro stahování webových stránek byl implementován distribuovaný systém, který využíval 35 počítačů. Ze stažených dat byly odstraněny duplicity. Vytvořené korpusy byly vzájemně porovnány pomocí různých statistik, jako jsou průměrná délky slov a vět, podmíněná entropie a podmíněná perplexita.

Klíčová slova: jazykový korpus, distribuované zpracování

Title: Large Multilingual Corpus

Author: Martin Majliš

Department: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský Ph.D.

Abstract:

This thesis introduces the W2C Corpus which contains 97 languages with more than 10 million words for each of these languages, with the total size 10.5 billion words. The corpus was built by crawling the Internet. This work describes the methods and tools used for its construction. The complete process consisted of building an initial corpus from Wikipedia, developing a language recognizer for 122 languages, implementing a distributed system for crawling and parsing webpages and finally, the reduction of duplicities. A comparative analysis of the texts of Wikipedia and the Internet is provided at the end of this thesis. The analysis is based on basic statistics such as average word and sentence length, conditional entropy and perplexity.

Keywords: language corpus, distributed processing

Contents

1. Introduction

As statistical approaches become the dominant paradigm in natural language processing, there is an increasing demand for data. It is known that simple models and a lot of data outclass sophisticated models based on less data. The web contains huge amounts of linguistics data for many languages. The web has many undeniable advantages: (a) size — it is the largest text collection containing billions of documents and its size is exponentially growing, (b) range — texts are available in many languages, styles and domains, (c) availability — most of the documents are available in machine-readable form, so no scanning or rewriting is necessary.

One of the key issues for computational linguists is easy access to such data. These data are publicly available on the Internet, but already collected corpora are available only for the major word languages, but not for most of the other languages.

Therefore, my aim is to collect, with minimal or no human intervention, at least ten millions of words for as many languages as possible.

1.1 Problem Definition

The goal of this thesis is to build multilingual corpus of texts available on the Internet. This corpus will consist of at least 10 million words for as many languages as possible. The collected material will be quantitative, and qualitative, analysed and conclusions about different languages will be made.

The project consists of:

- A study of existing multilingual resources and approaches used to construct them.
- A review of tools and methods used for solving particular tasks such as building initial corpora, crawling, language recognition and duplicity detection.
- A design for solving these particular tasks as well as the main tasks with respect to amount of processed data.

- An implementation of tools and processes capable of taking benefits of distributed environment.
- A quantitative and qualitative analyses of the collected material.
- Conclusions about used methods with evaluation of their performances for different languages.

1.2 Motivation

There are many publicly available projects that are trying to collect multilingual textual resources. Some of them cover many of languages but contain either very few documents or these documents are not in computer accessible form, so they cannot be easily used in computational linguistics. Other projects contain more data, but are available in very few languages. Therefore, it will be useful to construct corpus, that will overcome these disadvantages. When this data becomes available, it will be possible to use it for comparative analysis of related languages, building language models for various applications such as machine translation, speech recognition, spell checking, etc. For achieving the main goal, many subtasks has to be solved, such as recognizing languages or downloading millions of web pages. When all this data is collected, it will be possible to use it for further improvements.

Apart from these objective motivations, there are also my personal motivations. Working on this project gives me a chance to get insight, knowledge and hands-on experience on processing massive amounts of data.

1.3 Thesis Organization

The work is divided into five chapters, beginning with the introductory Chapter 1 containing problem definition and motivation. Chapter 2 gives an overview of existing methods and techniques. It briefly introduces existing multilingual resources and multilingual corpora as well as methods used for their construction. It also presents methods for solving particular steps. Chaprter 3 presents requirements for the complete system and available computational resources. It also introduces implemented tools and methods how to use them effectively. Chapter 4 shows achieved results in language recognition and size of constructed corpus. A quantitative and qualitative analyses of the corpus is included. Chapter 5 discusses the results and areas where the methods and implementation could be improved. It also suggests goals for the for future work.

Four appendices are included: Appendix A describes the content of the DVD. Appendix B contains lists of languages covered by the collected corpus with their ISO-639-3 codes and. Appendix C presents differences between the Wiki Corpus and the W2C Corpus.

2. Literature Review

This chapter reviews existing tools, methods and approaches. It opens by presenting statistics about existing languages, followed by an introduction of existing multilingual projects and multilingual web corpora. The end of this chapter contains an overview of methods used for crawling, text extraction, language recognition and corpus storing and distribution.

2.1 Existing Languages

There are 6,909 known living languages according to the Ethnologue databse [1], but only about 390 of them are used by more than 1 million of native speakers[2], while 172 of them have more than 3 million speakers.

Detailed distribution of languages and speakers is showed in Table 2.1 and Figure 2.1. These numbers must be treated with caution, because they are slightly out-of-date. Total population according to this table is 6 billion but it was true in 1999[3].

According to Wikipedia, there are 116 official languages[4].

2.2 Language Resources

There are many projects aim to collect materials in as many languages as possible, because there are predictions, that fifty percent of the world's languages will disappear in the next century[5].

Following projects are reviewed:

- The Rosetta Project (2.2.1)

[1] `http://www.ethnologue.com/web.asp`

[2] `http://www.ethnologue.com/ethno_docs/distribution.asp?by=size`

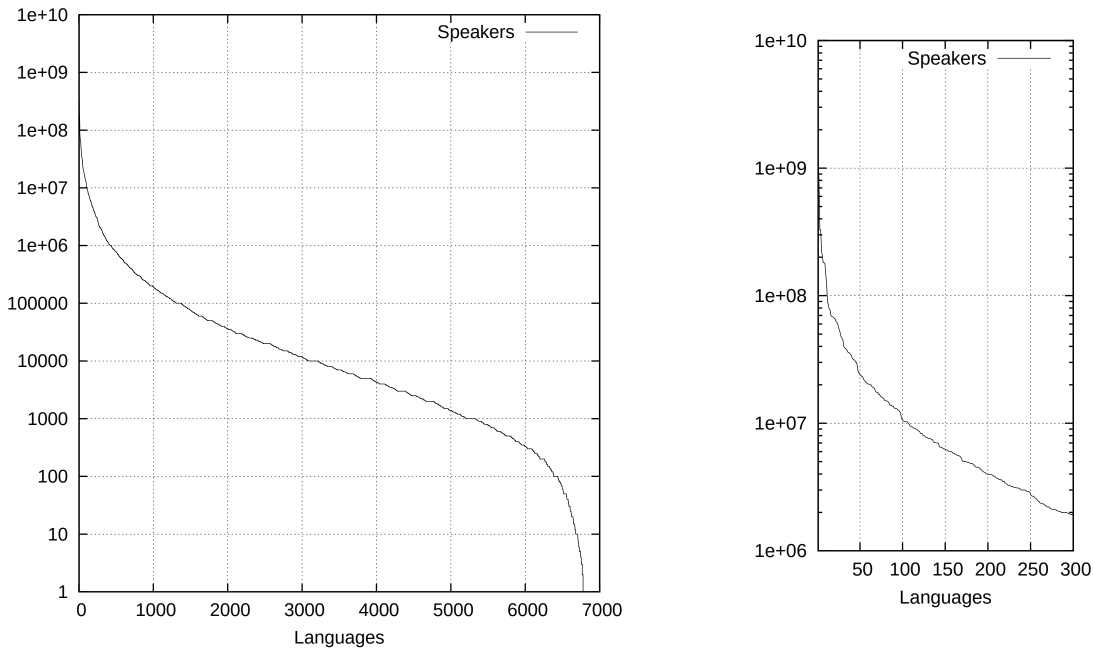
[3] `http://www.census.gov/population/international/data/idb/worldpopgraph.php`

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

[5] `http://www.unesco.org/new/en/culture/themes/cultural-diversity/languages-and-multilingualism/endangered-languages/`

Population range	Living languages			Number of speakers		
	Count	Percent	Cumulative	Count	Percent	Cumulative
100,000,000 to infinity	8	0.1	0.1%	2,308,548,848	38.73721	38.73721%
10,000,000 to 99,999,999	77	1.1	1.2%	2,346,900,757	39.38076	78.11797%
1,000,000 to 9,999,999	304	4.4	5.6%	951,916,458	15.97306	94.09103%
100,000 to 999,999	895	13.0	18.6%	283,116,716	4.75067	98.84170%
10,000 to 99,999	1,824	26.4	45.0%	60,780,797	1.01990	99.86160%
1,000 to 9,999	2,014	29.2	74.1%	7,773,810	0.13044	99.99204%
100 to 999	1,038	15.0	89.2%	461,250	0.00774	99.99978%
10 to 99	339	4.9	94.1%	12,560	0.00021	99.99999%
1 to 9	133	1.9	96.0%	521	0.00001	100.00000%
Unknown	277	4.0	100.0%			
Total	6,909	100.0		5,959,511,717	100.00000	

Table 2.1: Distribution of languages by number of first-language speakers



(a) All Languages (b) Top 300 Languages

Figure 2.1: Distribution of languages by number of first-language speakers

- The Open Language Archives Community (2.2.2)
- The Wikipedia (2.2.3)
- The Universal Declaration of Human Rights (2.2.4)
- The Project Gutenberg (2.2.5)
- The Wikisource (2.2.6)
- The Watchtower (2.2.7)
- Urbi et Orbi (2.2.8)
- Open-source Software (2.2.9)

2.2.1 Rosetta Project

The Rosetta[6] Project is a global collaboration of language specialists and native speakers working to build a publicly accessible digital library of material on all known human languages. The collection currently contains nearly 100,000 pages of material spanning over 2,500 languages, as well as a growing multimedia collection of modern and historical language recordings.

This material is publicly available on the Internet Archive website[7]. Most of the languages are covered very briefly. For example for Yami[8] with three thousand speakers there is only a dictionary[9]. For Czech[10] with twelve million speakers there is only a dictionary and the Universal Declaration of Human Rights[11].

The 300 Languages Project[12] is Rosetta's sub-project with a specfic goal of compiling a universal collection of 300 most widely spoken languages. This collection will contain parallel texts and recordings.

2.2.2 Open Language Archives Community

The Open Language Archives Community[13] (OLAC) is an international partnership of institutions and individuals who are creating a worldwide virtual library

[6] `http://rosettaproject.org/` and `http://www.archive.org/details/rosettaproject`

[7] `http://www.archive.org/browse.php?field=subject&mediatype=texts&collection=rosettaproject`

[8] `http://www.ethnologue.com/show_language.asp?code=tao`

[9] `http://www.archive.org/details/rosettaproject_tao_swadesh-1`

[10] `http://www.ethnologue.com/show_language.asp?code=ces`

[11] `http://www.archive.org/search.php?query=language\%3A\%22ces\%22`

[12] `http://rosettaproject.org/projects/300-languages/`

[13] `http://www.language-archives.org/`

Population range	Languages	Coverage			Online Resources		
		Count	Percent	Items	Count	Percent	Items
100,000,000 to 999,999,999	8	8	100%	7745	8	100%	1007
10,000,000 to 99,999,999	77	75	97%	4367	72	94%	2152
1,000,000 to 9,999,999	304	277	91%	4887	246	81%	3006
100,000 to 999,999	895	716	80%	8814	600	67%	4388
10,000 to 99,999	1824	1181	65%	15208	951	52%	5581
1,000 to 9,999	2014	1244	62%	20566	1097	54%	8190
100 to 999	1038	634	61%	11239	560	54%	3799
10 to 99	339	235	69%	6427	202	60%	1075
1 to 9	133	90	68%	1067	75	56%	519
Unknown	277	115	42%	1731	79	29%	394
All living languages	6909	4575	66%	82051	3890	56%	30111
Extinct languages	520	242	47%	2328	178	34%	778

Table 2.2: OLAC - language coverage

Articles	Count	Cumulative
1,000,000 to 9,999,999	3	3
100,000 to 999,999	34	37
10,000 to 99,999	64	101
1,000 to 9,999	107	208
100 to 999	60	268
10 to 99	7	275
1 to 9	5	280

Table 2.3: Wikipedia - article counts

of language resources. Their language coverage is presented in Table 2.2.

2.2.3 Wikipedia

Wikipedia[14] is a free, web-based, collaborative, multilingual encyclopedia project. It contains 19 million articles in 281 languages[15]. Article counts are presented in Table 2.3.

[14] http://www.wikipedia.org/

[15] http://meta.wikimedia.org/wiki/List_of_Wikipedias

2.2.4 Universal Declaration of Human Rights

The Universal Declaration of Human Rights[16] (UDHR) is a milestone document in the history of human rights. At present, there are 379 different translations of UDHR, available in HTML and/or PDF format. This project sets the Guinness World Record for Most Translated Document[17].

There is a related project UDHR in Unicode[18] which aims to convert all documents into Unicode, although only four of them have been completed and reviewed[19].

2.2.5 Project Gutenberg

The Project Gutenberg[20] is a volunteer effort to digitize and archive cultural works. It contains over 34 thousands documents in 60 languages. Most of the items are the full texts of public domain books.

2.2.6 Wikisource

Wikisource[21] is an online library of free content textual sources, operated by the Wikimedia Foundation. Its aims are to harbour all forms of free text, in many languages. Wikisource contains more than 1M articles in 62 languages[22].

2.2.7 Watchtower

The Watchtower[23] is an illustrated religious magazine, published semi-monthly by Jehovah's Witnesses. It is written in 418 languages (366 without sign languages). Texts are available as web pages or PDF files. All files have a very similar structure, so it may serve as a very good source of parallel texts.

[16] http://www.ohchr.org/EN/UDHR/Pages/Introduction.aspx
[17] http://www.ohchr.org/EN/UDHR/Pages/WorldRecord.aspx
[18] http://unicode.org/udhr/
[19] http://unicode.org/udhr/index_by_stage.html
[20] http://www.gutenberg.org/
[21] http://www.wikisource.org/
[22] http://meta.wikimedia.org/wiki/Wikisource\#List_of_Wikisources
[23] http://watchtower.org/

2.2.8 Urbi et Orbi

Urbi et Orbi[24] is the blessing which takes place at each Easter and Christmas celebration in Rome from the central loggia of St. Peter's Basilica, at noon. This year (2011) was pronounced in 65 languages, the highest number in the history. This blessing consists of a single sentence: "May the grace and joy of the Risen Christ be with you all."

2.2.9 Open-source Software

Open-source Software[25] is computer software that is available in source code typically developed by volunteers distributed amongst different geographic regions. Therefore, big OSS projects are available in many languages. These string are mostly texts of error messages, menus and buttons. For example:

- Launchpad[26] - 323 languages, 1,730,838 strings
- Gnome[27] - 173 languages
- KDE[28] - 75 languages

2.2.10 Summary

Sizes of different language resources are summarized in Table 2.4. From these sizes, it is possible to conclude:

- Thousands of languages are available in the Rosetta Project and the Open Language Archives Community. To achieve this, special language interest groups and linguistics specialists are required.
- Around 300 languages are presented in the Universal Declaration of Human Rights, Wikipedia, the Watchtower and Launchpad. This is the upper bound for number of languages that are at least theoretically available in written form on the Internet. This covers almost 90% of all people.

[24] http://en.wikipedia.org/wiki/Urbi_et_Orbi
[25] http://en.wikipedia.org/wiki/Open_source_software
[26] https://translations.launchpad.net
[27] http://l10n.gnome.org/languages/
[28] http://l10n.kde.org/teams-list.php

Projects	Languages	Size
Rosetta Project 2.2.1	over 2,500	100,000 pages
OLAC 2.2.2	4,575	82,051 items
Wikipedia 2.2.3	281	19,034,746 articles
UDHR 2.2.4	379	at most 379 documents
Project Gutenberg 2.2.5	60	34,000 documents
Wikisource 2.2.6	62	1,028,303 pages
Watchtower 2.2.7	366	thousands of pages
Urbi at Orbi 2.2.8	65	65 sentences
Launchpad 2.2.9	323	1,730,838 strings
Gnome 2.2.9	173	about 1 million of strings

Table 2.4: Multilingual resources — summary

- Around 60 languages are available in Project Gutenberg, Wikisource and Urbi at Orbi. This is the lower bound for the number of languages that are used in developed or newly industrialized countries[29] countries. This covers almost 70% of all people.

2.3 Multilingual Web Corpora

As early as 2001, Banko and Brill [BB01] and recently in 2009 Halevy et al. [HNP09], showed that using more data and simple method outperform less data and sophisticated method.

The following multilingual web corpora WaCky (2.3.1), Crúbadán (2.3.2), I-X (2.3.3) and Corpus Factory (2.3.4) are reviewed in more details. The unit 'W' will be used instead of word, so 10MW means 10 million words.

2.3.1 WaCky

WaCky was introduced for the first time in Baroni and Kilgarriff [BK06] in 2006 with more detailed information in [BBFZ09]. This corpus contains 3 languages - English, German and Italian — and each of them has approximately 1.5TW.

They randomly combined mid-frequency content words from existing corpora for each language to construct word pairs. This bigrams were used for constructing search queries for Google to retrieve a list of seed URLs.

[29] http://en.wikipedia.org/wiki/Newly_industrialized_country

Property	deWaC	itWaC	ukWac
Raw crawl size (GB)	398	379	351
Documents after filtering (M)	4.86	4.43	5.69
Size after document filtering (GB)	20	19	19
Size after near-duplicate cleaning (GB)	13	10	12
Documents after near-duplicate cleaning (M)	1.75	1.87	2.69
Tokens (G)	1,278	1,586	1,914

Table 2.5: WaCky — data size

Heritrix[30] was used to crawl pages with breadth-first crawling strategy. Crawling was restricted to pages in relevant web domains (.de/.at for German; .it for Italian; .uk for English). URLs with suffix indicating non-HTML data (.pdf, .jpg, etc.) were discarded

From the Heritrix log file they retrieved pages with mime type text/html and between 5 and 200kB. These pages were preserved. For removing boilerplate code they used their own reimplementation of the BTE tool[31].

The cleaned documents were filtered based on lists of function words. Documents not meeting minimal requirements — 10 types and 30 tokens per page, with function words accounting for at least a quarter of all words were discarded. This filter also worked as a simple language identifier. They also used a blacklist to discard pornographic pages.

Near duplicate detection was performed by a simplified version of Broder's "shingling" algorithm [BGMZ97]. They removed functional words and randomly selected 25 5-grams from each document. If a pair of documents shared at least two 5-grams, they were considered as near duplicate and one of them was removed.

Building a corpus for each language took approximately 3 weeks (10 days crawling, 7 days cleaning, 4 days near-duplicate detection). Basic statistics are presented in Table 2.5.

[30] `http://crawler.archive.org/`

[31] `http://dev.sslmit.unibo.it/wac/post_processing.php`

2.3.2 Crúbadán

Crúbadán is a multilingual corpus introduced by Scannel [Sca07]. This corpus contains 487 languages[32].

For each language, some additional metadata was provided manually: the name of the language in English, the ISO 639-3 code, a flag indicating whether the language is under-resourced, and a list of "polluting" languages (languages frequently used in boilerplate texts).

The majority of training texts came from three sources: Wikipedia, the Watchtower (Jehovah's Witnesses web site) and the Universal Declaration of Human Rights site. Training texts were preprocessed. A temporary word frequency list was generated and then several filters were applied to produce a clean word list. For example the following words were removed: words with characters not usually appearing in the target language, words with no vowels (when this made sense), words with the same character appearing three or more times in a row, words with a capital character appearing after the first character, words that appeared in the word list for a polluting language, and words that contained improbable letter trigrams (at later stages, after the statistics were available). Native speakers were asked to define language specific constrains.

Stop words were extracted by native speakers. When no native speaker was available, the highest frequency words that did not appear as high frequency words in other languages were used.

Search queries were generated by combining randomly chosen words, connected by OR and at least one stop word connected by AND. They used Google to retrieve URLs and `wget`. For the conversion to plain text, they have used the open source programs `vilistextum`, `pdftotext` and `wvText`.

Language detection is based on comparing the cosine of the angle between vectors representing downloaded document and training documents in the space of character trigrams with manual tuning based on a number of ad-hoc factors.

Crúbadán corpus size is presented in Table 2.6.

Scannel also states:

> Indeed, we claim that any effort to crawl the web for a large num-

[32] `http://borel.slu.edu/crubadan/stadas.html`

(a) Document counts

Document count	Languages
> 1k	70
> 500	115
> 250	143
> 125	181
> 65	210
> 32	255
> 16	337
> 8	356
> 4	381
> 2	416
> 1	449

(b) Word counts

Word count	Languages
> 100MW	1
> 10MW	11
> 1MW	127
> 100kW	225
> 10kW	354
> 1kW	473
> 100W	487

Table 2.6: Crúbadán — data size

ber of languages without attempting to harness the collective knowledge of many language experts, ..., is doomed to failure.

2.3.3 I-X

Sharoff [Sha06] introduced BNC-like multilingual web corpus. This corpus contains 6 languages — English, German, Russian, Chinese, Romanian and Ukrainian, but only for three of them are results available.

500 common words were chosen from existing corpora and constructed queries by combining N-tuples (N = 2-4) of such words connected by AND and prefixed by 2 very frequent words connected by OR.

5,000 queries were used and with Google API, 50,000 URLs were retrieved. These URLs were downloaded without recursion. Encoding was unified and lynx[33] was used to convert pages from HTML to plain text (worked better than ad-hoc Perl filters). Then, simple heuristic was used for navigation frame detection (links density). For deduplication they used a simplified version of "shingling" algorithm from WaCky (2.3.1).

The corpus size is presented in Table 2.7. The corpora for Chinese, Romanian and Ukrainian are mentioned only in the introduction and no results are presented.

[33] `http://lynx.isc.org/` - text web browser

Language	Size in MW
English (I-EN)	127
German (I-DE)	126
Russian (I-RU)	156
Chinese	???
Romanian	???
Ukrainian	???

Table 2.7: I-X — size in MW

2.3.4 Corpus Factory

Corpus Factory is a multilingual corpus constructed by Kilgarriff [KRPA10]. This corpus contains 8 languages - Dutch, Hindi, Indonesian, Norwegian, Swedish, Telugu, Thai and Vietnamese.

Firstly, they built corpora from Wikipedia pages with at least 500 words (Wiki Corpora). Secondly, they tokenized these corpora. For languages with absent explicit word delimiters, (Thai, Vietnamese) they used language specific tools and space and punctuations marks for the rest.

They considered the top 1000 words as high-frequency words and the next 5000 words as mid-frequency ones. They used only words with at least 5 characters (except for Vietnamese, where words may contain spaces).

They used BootCaT's query generation module. The number of words in a query was language dependent and was automatically assigned. They also found out, that Google normalizes many non-UTF8 encodings to UTF-8 whereas Yahoo and Bing don't. For licensing and usability reasons they used Yahoo and Bing, therefore they converted UTF8 word seeds into native encodings.

`Wget` was used for downloading web pages and only pages with mime type text/HTML and size between 5kB and 2MB (this information was provided by the search engine API).

They used BTE[34] algorithm to remove boilerplate code and to retrieve plain text. They considered 500 words with the highest frequency (from Wiki Corpora) as functional words. Then they sorted wiki pages according to the proportion of top-500 words. They found out, that the top 70% of pages contains connected texts. These pages contain at least 65% of the words from the top-500 words.

[34]Body Text Extraction algorithm (BTE, Finn et al. 2001)

Language	Wiki Corpora	Web Corpora
Dutch	30.0	108.6
Hindi	2.5	30.6
Indonesian	8.5	102.0
Norwegian	19.1	94.9
Swedish	9.3	114.0
Telugu	0.2	3.4
Thai	6.2	81.8
Vietnamese	9.5	149.0

Table 2.8: Corpus Factory — size in MW

Therefore they preserved only web pages with at least 65% of the words from the top-500.

For near-duplicate detection, they used perl's Text::DeDupper module which implements Broder's "shingling" algorithm.

Corpora size is displayed in Table 2.8.

2.3.5 Summary

In this subsection, I summarize existing multilingual corpora and compare them with one another. Sizes are presented in Table 2.9.

All approaches used very similar methods:

1. Retrieve word seeds from existing corpora or reliable text source.
2. Generate n-tuples of words.
3. Use these tuples as search queries.
4. Download found web pages.
5. Preserve just files with mime text/html and acceptable size.
6. Use BTE for removing boilerplate code.
7. Use functional words for language detection and running text detection.
8. Use Broder's "shingling" algorithm to find near duplicate detection.

Differences among all approaches are displayed in Table 2.10.

Language	WaCky	Crúbadán	I-X	Corpus Factory
English	1,914GW	26.8MW	127MW	No
German	1,278GW	2.7MW	126MW	No
Russian	No	333kW	156MW	No
Italian	1,586GW	3.2MW	No	No
Dutch	No	2.6MW	No	138.6MW
Hindi	No	805kW	No	33.1MW
Indonesian	No	5MW	No	110.5MW
Norwegian	No	1.3MW (B), 2.6MW (N)	No	114MW
Swedish	No	2MW	No	123.3MW
Telugu	No	2MW	No	3.6MW
Thai	No	218kW	No	90MW
Vietnamese	No	3.9MW	No	158.5MW
Chinese	No	320kW	Yes	No
Romanian	No	6.6MW	Yes	No
Ukrainian	No	273kW	Yes	No

Table 2.9: Language coverage

2.3.6 Conclusions

If I evaluate these projects with respect to the goals of this thesis, then size 10MW was fulfilled by 11 languages in Crúbadán, 7 in the Corpus Factory and 3 in WaCky and I-X. Methods used for building Crúbadán required native speakers or were computationally ineffective — language detection done by comparison with with all testing documents.

2.4 Word Seeds

Word seed is an initial small corpus, that is used as a source of words for generating queries, recognizing languages and estimating document quality. All multilingual web corpora (2.3) were using any existing reliable language resource as the initial corpus. They used Wikipedia (2.2.3) or established corpora such as the British National Corpus.

Property	WaCky	Crúbadán	I-X	Corpus Factory
Word seeds	Texts from existing corpora.	Texts from from specified website.	Texts from existing corpora.	Texts from Wikipedia.
URL seeds	Searching pairs of mid-frequency content words using google.	Searching randomly chosen words from lexicon (OR'ed together) with AND'ed at least one stopword.	Searching randomly chosen words from lexicon (AND'ed together) with OR'ed 2 high frequency words.	Searching mid-frequency words. Number of words is language dependent.
Crawler	Heritrix	wget	Unspecified	wget
Crawling	Domain restricted, suffix restricted. Recursive.	Extracted URLs are added to the pending list of URLs for the language of the downloaded document. Recursive.	Just extracted URLs. Without recursion.	Just extracted URLs. Without recursion.
Filtering	Mime type text/html, size between 5kB and 200kB.	Unmentioned	Unmentioned	Mime type text/html, size between 5kB and 2MB. At least 65% of high frequency words.
Boilerplate	Modified BTE algorithm.	Unmentioned	Tag density (maybe BTE)	BTE algorithm.
Deduplication	Simplified version of Broder's "shingling" algorithm.	Unspecified	Simplified version of Broder's "shingling" algorithm.	Broder's "shingling" algorithm.
Language Detection	Contains functional words.	Cosine angle between vectors representing the document and training texts in the space of character trigrams. Manual tuning.	Unmentioned. Functional words in search query.	Unmentioned. Functional words in search query.
Languages	3	487	3 (6)	8
Median size	1.586GW	68,221W	126MW	102MW

Table 2.10: Existing multilingual corpora — overview

2.5 URL Seeds

URL seeds is an initial list of URLs for crawler. There are at least 2 possible approaches to how to construct the list: use any existing list of URLs (2.5.1) or retrieve these URLs a from search engine (2.5.2).

2.5.1 Existing Lists

A web directory or a language resource with external links can be a good source of URLs in a specific language.

Open Directory Project

The Open Directory Project[35] (ODP or Dmoz) is a multilingual open content directory of World Wide Web links. It contains links to websites in 90 languages[36], although from 2006 it is no longer expanding[37].

Wikipedia

Wikipedia contains a lot of links to external web pages (links are mostly in the External Links section). It is also possible to retrieve all external links in single file - for example English[38].

2.5.2 Search Engines

Another approach is to use search engines to retrieve URLs. This method is used by all multilingual web corpora (2.3). They differ in the way they generate queries from word seeds - how much words should be used, which words should be chosen and how they should be combined.

[35] `http://www.dmoz.org/`

[36] `http://www.dmoz.org/World/`

[37] `http://commons.wikimedia.org/wiki/File:Odp_sitecount_top.png`

[38] `http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-externallinks.sql.gz`

Google Search

Google Search[39] is a web search engine owned by Google Inc. and is the most-used search engine on the Web.

In the past, there was Google Web Search API[40] but it has been officially deprecated as of November 1, 2010. There is also Custom Search API[41], which allows 100 queries per day and additional ones must be bought.

Bing

Bing [42] is a web search engine owned by Microsoft Corporation and is one of the most used search engine on the Web.

It provides API[43] for searching. The only limitation is less than 7 queries per second.

2.6 Crawling

There are plenty of crawlers available on the Internet. Some of them are:

GNU Wget

GNU Wget[44] is part of the GNU Project and is therefore is available on all linux machines.

- Very simple and easy to use.
- It can store HTTP headers.
- It is not possible to create rules that decides, whether to continue or terminate downloading according to HTTP header.
- A lot of websites returns different content or 4XX HTTP status code. It is possible to change the user agent.

[39] `http://www.google.com`
[40] `http://code.google.com/intl/cs/apis/websearch/`
[41] `http://code.google.com/apis/customsearch/v1/overview.html`
[42] `http://www.bing.com/`
[43] `http://www.bing.com/developers/`
[44] `http://www.gnu.org/software/wget/`

Nutch

Nutch[45] is a crawler that is built on Apache Lucene.

- It can run on single machine, but also on Hadoop[46] cluster.
- It supports plugins.

Heritrix

Heritrix[47] is a crawler developed by Internet Archive for web archiving.

- It is a very complex software with dozens of options.
- It can be very precisely tuned and missing functionality may be implemented as a plug-in.

2.7 Language Recognition

Language detection is one of the crucial parts of this project. This field has been researched since 1970s[48]. There are many articles about language recognition, but I found out that algorithms used in real applications are different. Therefore I at first introduce theoretical approaches (2.7.1) and then approaches used in some applications (2.7.2).

2.7.1 Theoretical

Cavnar and Trenkle [CT94] algorithm uses a sliding window over a set of characters. A list of the 300 most common n-grams for n in 1..5 is created during training for each training document . To classify the new document, they constructed the list of the 300 most common n-grams and compare n-grams position with testing lists. The list with minimal differences is the most similar one and new document is in same language. They were classifying 3478 samples in 14

[45] `http://nutch.apache.org/` and `http://en.wikipedia.org/wiki/Nutch`
[46] `http://hadoop.apache.org/`
[47] `http://crawler.archive.org/` and `http://en.wikipedia.org/wiki/Heritrix`
[48] `http://speech.inesc.pt/~dcaseiro/html/bibliografia.html`

languages from a newsgroup. They reported they achieved an accuracy of 99.8% (only 7 document were wrong).

Sibun [SR96] introduced a method for language detection based on relative entropy, a well-known measure also known as Kullback-Leibler distance. The relative entropy is a useful measure of the similarity between probability distributions. She used texts in 18 languages from European Corpus Initiative CD-ROM. She achieved accuracy for bigrams 100%.

Hayati [Hay04] reported, that with Cavnar and Trenkle's algorithm they achieved an 86.8% accuracy on webpages spanning 11 languages. Therefore, they used the Fisher discriminant function to choose representative n-grams for all the languages, and compared the new document to the reference using cosine similarity measure. Using this method, they achieved an accuracy of 93.9%. They also showed, that using information about incoming and outgoing links (webpages usually links to pages in the same language), increases the accuracy of the classifier.

Martins et. al [MS05] reported that with Cavnar and Trenkle's algorithm of improved metric for comparing lists of n-grams, they achieved accuracy 91.25% for 12 languages.

2.7.2 Applications

There are plenty of applications that use language detection and some of them have an accessible source code which is why I take a closer look at them.

Mozilla Firefox

Mozilla Firefox[49] is a free and open source web browser. Mozilla currently contains two charset detectors[50].

Chardet[51] is an original Mozilla project that is still used for charset detection. It uses precomputed bi-gram models. The source code is originally in C++[52] but

[49]`http://www.firefox.com/`
[50]`http://www.mozilla.org/projects/intl/chardet.html`
[51]`http://www-archive.mozilla.org/projects/intl/ChardetInterface.htm`
[52]`http://mxr.mozilla.org/seamonkey/source/intl/chardet/`

there are ports in Java[53] or Python[54].

Li and Momoi [LM01] universal charset detector[55] uses a combined approach. In the first phase, the code scheme gets checked. Some byte sequences are illegal in some encoding, so this is very effective for 7-bit multi-byte encodings. If encoding is not recognized then unigram distribution is used to detect encoding. If the detector is still not confident enough it will use bigram distribution. Source code are in C++ and are publicly available[56].

Google CLD

Google CLD[57] (Compact Language Detection Library) is a part of Google Chrome[58]. Google Chrome is also a free and open source web browser.

The CLD looks up each quadgram in a large hashtable that contains language probabilities. This hashtable was originally built by processing language probabilities over billions of web pages that are indexed by Google's search engine. The CDL is able to recognize approximately 90 languages[59].

The algorithm itself[60] uses informations from TLD - it is more probable that a page in TLD .cz will be in Czech than in Slovak). It also uses page encoding - windows-1250 is central European and therefore it will not be in any Asian language. It iterates over all quadgrams (HTML markup is ignored) and accumulates a score for each language. It is using information about language close pairs to modify the overall score. Language scores are also normalized by average score retrieved per kilobyte.

It also contains manually written rules. If the text is in English or language X (X is high enough), then assumes the English is boilerplate and the page is in language X. If the text is in FIGS (French, Italian, German or Spanish) or X (X

[53] http://jchardet.sourceforge.net/

[54] http://chardet.feedparser.org/

[55] http://www.mozilla.org/projects/intl/UniversalCharsetDetection.html

[56] http://mxr.mozilla.org/seamonkey/source/extensions/universalchardet/

[57] http://googletranslate.blogspot.com/2010/03/faster-simpler-and-safer-browser-goes.html

[58] http://www.google.com/chrome/

[59] http://src.chromium.org/viewvc/chrome/trunk/src/third_party/cld/languages/proto/languages.pb.h?view=markup

[60] http://src.chromium.org/viewvc/chrome/trunk/src/third_party/cld/encodings/compact_lang_det/compact_lang_det_impl.cc?view=markup

is not English and is high enough), then it assumes the FIGS is boilerplate and page is in language X.

There is a lot of magic numbers for different thresholds, ratios, etc.. Hashtables of quadgrams are declared in this file[61]. Code also contains interesting comments[62]:

> Restrict the set of scored languages to the Google "Top 40*", which is actually 38 languages. This gets rid of about 110 language that represent about 0.7% of the web. Typically used when the first pass got unreliable results.

Google Translate API

Google Translate API[63] is service provided by Google to translate texts between 52 languages. It has also an interface for language detection[64]. It is very probable, that it is using Google CLD.

This API was officially deprecated as of 26th May 2011 and will be shut down on 1st December 2011[65].

2.7.3 Multilingual Web Corpora

Multilingual web corpora use different methods (summarized in Table 2.10). WaCky detects functional words in document. I-X and Corpus Factory rely on functional words in search queries. Crúbadán compares the cosine angle between vectors representing the document and training texts in the space of character trigrams with manual tuning.

[61] `http://src.chromium.org/viewvc/chrome/trunk/src/third_party/cld/encodings/compact_lang_det/generated/compact_lang_det_generated_quads_256.cc?view=markup`

[62] `http://src.chromium.org/viewvc/chrome/trunk/src/third_party/cld/encodings/compact_lang_det/compact_lang_det_impl.h?view=markup`

[63] `code.google.com/apis/language/translate/overview.html`

[64] `http://code.google.com/intl/cs/apis/language/translate/v2/using_rest.html\#detect-language`

[65] `http://googlecode.blogspot.com/2011/05/spring-cleaning-for-some-of-our-apis.html`

Paper	Languages	Accuracy
Cavnar and Trenkle [CT94]	14	99.8%
Sibun [SR96]	18	100%
Hayati [Hay04]	11	93.9%
Martins et. al [MS05]	12	91.25%
Google CLD 2.7.2	87	unknown
Google Translate API 2.7.2	53	unknown
WcCky 2.3.1	3	unknown
Crúbadán 2.3.2	489	unknown
I-X 2.3.3	3	unknown
Corpus Factory 2.3.4	8	unknown

Table 2.11: Language detection — summary

2.7.4 Summary

An overview of all these papers is in Table 2.11. The Crúbadán has the highest number of recognized languages, but more than 100 languages contains four or less documents and half of them less than 40, so I suppose that a lot of them were assigned manually.

2.8 Corpus Storing and Distribution

Corpus storing and distribution is one of the fundamental parts of corpus building. Wynne ([Wyn05]) as well as E-MELD[66] suggests many tips.

Archival copies should be made in a format which offers LOTS (i.e., it is Lossless, Open Standard, Transparent, and Supported by multiple vendors). A corpus must also contain proper documentation of used formats along with information about terms of use, and access restrictions.

Making a corpus widely available should not be possible due to copyright and other legal issues.

[66] http://emeld.org/school/bpnutshell.html

2.9 Corpus Quality Analysis

Corpus analysis is an important step in building web corpus. Without comparing with existing corpora it is hard to say whether high quality texts were downloaded or if they are just some 'CD image'.

Rayson et. al [RG00] suggested using log-likelihood statistics for comparing frequency lists. This approach was used in all multilingual corpora. Bharati et. al [BRSB00] also suggested using a number of unique unigrams, entropy, word and sentence lengths for comparing different corpora.

2.10 Internet Size

When the corpus is downloaded, it is useful to know, how much can it be extended. In 1997, Bharat et al. [BB98] used 300,000 documents in the Yahoo! hierarchy to build a lexicon of about 400,000 words (low frequency words were excluded). Then they constructed random queries and retrieved random pages from first 100 results. They used 35,000 queries and 4 search engines to estimate the size of the Internet in November 1997 which was at least 200 million pages.

In 2005, Gulli et al. [GS05] used a very similar method but in larger scale. They used 438,141 queries in 75 different languages. They also used four search engines and they found out, that their overlap is just 28.85%. They needed 43 Linux servers, requiring about 70Gb of bandwidth and more than 3600 machine-hours. They estimated that the indexable web has more than 11.5 billion pages.

Broder et al.[BFJ$^+$06] and Bar-Yossef et al. [BYG07] showed that random queries do not return random documents and this is causing an underestimation of the the size. To overcome this problem, Lu et al. [LL10] introduced estimator based on the capture–recapture methods.

From the other resources, there are more than 255 million websites[67] and almost [68] 50 billion webpages. The former Google CEO, Eric Schmidt, states in 2005, that Google is indexing 170GB[69]. The search engine Cuil[70] indexed more than 121 billion pages in 2007.

[67] `http://www.focus.com/images/view/48564/`
[68] `http://www.worldwidewebsize.com/`
[69] `http://news.softpedia.com/news/How-Big-Is-the-Internet-10177.shtml`
[70] `http://en.wikipedia.org/wiki/Cuil`

The Internet has now more than 2.1 billion users[71], doubled since 2007. Zhang et al. [ZZY+08] showed that the number autonomous system doubles every 5.3 years.

[71] `http://www.internetworldstats.com/emarketing.htm`

3. Methods

This chapter describes tools and methods used for building web corpus. Complete process is illustrated on Figure 3.1 with available resources and data flow.

Constructing of web corpus consists of several step. The initial step was gathering metadata from Wikipedia and Ethnologue. The downloaded metadata are stored into the database on the hosting. When matadata was available, then Wiki Corpus was built from Wikipedia articles. Frequency lists for trigrams and quadgrams were computed and uploaded to the hosting. From the Wiki Corpus the language model was trained and moved to the hosting. Building web corpus was divided smaller jobs, that were executed in the computer laboratory. Job results were stored on ufallab, where they were merged into raw corpus. This raw corpus was transferred back to the laboratory, when duplicity was reduced and statistics were computed. The clean corpus was stored on ufallab.

3.1 Available Resources

I had access to the following computational resources during my work on this thesis.

- PC (stingray) — Intel Pentium 4 CPU 1.80GHz, 2GB RAM, 230GB disk
- Computer laboratory[72] — available for all students of our faculty — 15 computers with Intel Core i7 920 (4x2.67GHz + HyperThreading), 6GB

[72] `http://wiki.ms.mff.cuni.cz/wiki/Po\%C4\%8D\%C3\%ADta\%C4\%8De_UNIX`

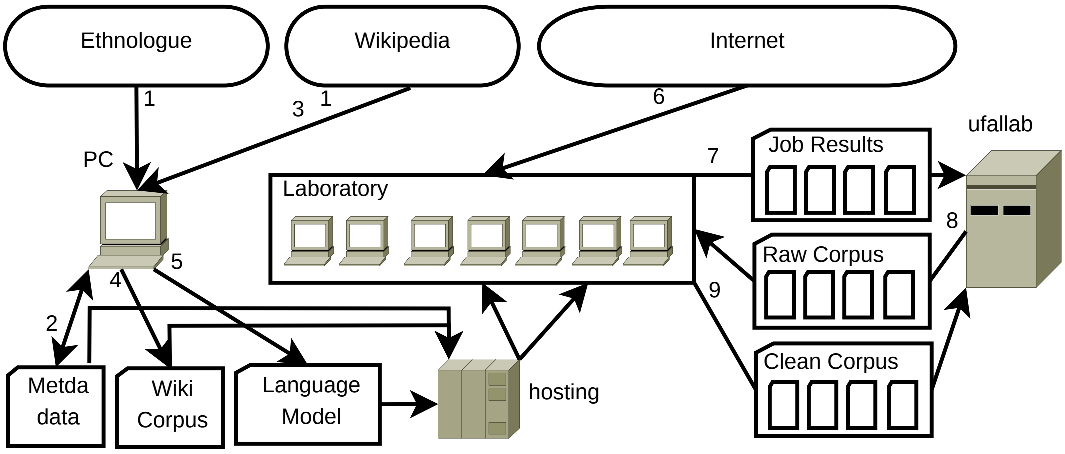


Figure 3.1: Building Web Corpus

RAM, 150GB disk space, 16 computers with Intel Core2 Quad Q9550 (4x2.83GHz), 4GB RAM and 50GB disk space, 6 computers with AMD 64 X2 3800+/4200+ (2x2GHz/2.2GHz), 2GB RAM and 50GB disk space and 5GB on shared network disk.

- Server (ufallab) — Intel Pentium 4 CPU 3GHz, 2GB RAM, 1.1TB disk
- Hosting — shared webhosting, 50GB disk space

The most serious limitation of the used resources was absence of possibility to establish ssh connection between the computer laboratory and ufallab without manually typed password. This password typing would be required every 15 minutes and more than two thousand times, if n other solution would be found.

Another complication, that I strongly underestimate in early stages, was very unpredictable environment in the computer laboratory. The main complications were:

1. Very low (for my needs) quotas on network traffic. I was able to consume weekly quota (2GB of outgoing traffic) in less than two hours. I negotiated disabling this quotas.
2. Any program can not run longer than 24 hours, so I have to divide work into smaller jobs.
3. This laboratory is used by many students and quite often some of them executed programs that consumed all memory or worked as fork bombs[73]. This behaviour has two consequences. The first one was, that any running program could crash, because it could not execute any subprogram. I added scripts that were restarting crashed programs, but these programs were also crashing, so I monitored them manually. The second one was, that the stacked computer might be restarted (by student or laboratory service). In this situation I was trying to use as much as possible from already processed data.
4. There is shared file system used for home directories in this laboratory. So very few programs intensively interacting with file system are causing lags (in seconds, dozens such programs can lag longer than a minute). These situations induces unpredictable behaviour of file operations. For example two scripts executed in serial order, where the first one creates a file and the second reads it, can cause, that the file does not exist, when the second script is executed.

[73] http://en.wikipedia.org/wiki/Fork_bomb

5. Problems mentioned in points 3 and 4 have also social aspect. If computers are working slowly, then users start complaining to the laboratory service or administrator. It is easy for them to expect, that the users with more than 2 thousands running processes, is causing problems. I had to defend myself to avoid account deletion.

3.2 General Principles

The design was based on maximal utilization of available resources (3.1) and their limitations. I decided to use many small components, that could be easily connected into more complex components. I also preferred to start with small scale experiments and simple scripts to explore the reality.

3.2.1 Usability

The usability is important for product spreading among users. It is very frustrating, when a program is executed and nothing happens or when it crashes with a cryptic message, because another required program or library is missing.

To overcome these problems, all scripts check fulfilling of their own requirements and additionally, there is a script `checkRequirements.sh` checking requirements of all scripts.

All scripts also use special notation for writing comments, that easily allows to automatically generate help if parameter `-h` is used and generate HTML documentation[74].

3.2.2 External Tools

I decided to employ as many already existing open-source software components as possible. External tool may be widespread and therefore may be included in an OS distribution package repository and easily installable (with `apt`, `yum`, etc.), or it may be a specific tool that must be retrieved from developers website.

There are at least 3 ways how these dependencies can managed:

[74] `http://w2c.martin.majlis.cz/w2c/doc-gen/`

1. They are just mentioned in a documentation.
2. They are bundled with a project.
3. They are retrieved from an external resource.

All these approaches have their pros and cons. The first one is the easiest one for the developer. If the external tool is widespread, then this possibility is also very convenient for the user. The second possibility gives the developer high control over the execution environment but for the cost of expanding project size. The third one haves benefits from the second one (control over environment) and it also does not increase project size.

I decided to use the third one, because it provides many benefits for users. As I mentioned in the section about usability (3.2.1), there is a script `checkRequirements.sh` that checks all requirements of external tools and libraries. If any of them is missing, then tips for installation are provided.

I also found out, that there are huge differences in the performance of different versions of the same software. For example grep 2.5.4 (that was available in the computer laboratory) is in some situations 60 times slower than version 2.6.3 or newer. Filtering 2 million lines with grep 2.5.4 took more than 13 minutes but with 2.6.3 took slightly more than 12 seconds. Therefore also program versions are checked and newer ones are installed.

3.3 Metadata

Metadata, such as language name, its ISO code, population size, writing system, etc., was for each language automatically downloaded from the Internet. The Following sources were combined:

- SIL International[75] — which provides easily parsable table[76] of all languages with their ISO codes and names.
- Wikipedia[77] — with its list of all wikipedias[78], where they use their own codes and names.

[75] `http://sil.org`
[76] `http://www.sil.org/iso639-3/iso-639-3_20100707.tab`
[77] `http://www.wikipedia.org/`
[78] `http://meta.wikimedia.org/wiki/List_of_Wikipedias`

- Ethnologue[79] — with easily parsable pages with language information - e.g. Czech[80].

Because I knew that the Ethnologue numbers are out-of-date (2.1), I intended to use information from the info-boxes in Wikipedia. For example, English has 328 million speakers according to Ethnologue[81], while Wikipedia[82] provides also information about first and second language speakers with overall up to 1.8 billion speakers. In fact, English is the 'Lingua franca' of the Internet therefore I would prefer to use numbers from Wikipedia.

To avoid parsing Wikipedia, I wanted to use DBpedia[83], which extracts information from Wikipedia, but I discovered that it is not reliable. For example, for the Buginese language DBPedia[84]: 240 speakers, Wikipedia:[85]: 3.5 to 4 millions and Ethnologue [86]: 3.5 millions.

From this I concluded, that information extraction from Wikipedia may not be easy. Not all languages are present and it may be hard to localize them, due to their name variants. It would be also hard to automatically and correctly decide, which number of speakers is correct. Therefore, I decided to stick with Ethnologue.

Scripts used for metadata extraction are `langList.sh` and `ethnologueParser.sh`.

In the early stages, extracted information was stored in text files. Later on, they were moved into a database (3.4).

3.4 Database

The database is used for storing metadata (3.3) and achieved results. In the early stages I was using text files but that required synchronization (among nb, pc, lab, ufallab), so I decided to use a database. I wanted to use an key-value store[87]. Due to limitations, I could not install anything on my computer that

[79] `http://www.ethnologue.com/`
[80] `http://www.ethnologue.com/show_language.asp?code=ces`
[81] `http://www.ethnologue.com/show_language.asp?code=eng`
[82] `http://en.wikipedia.org/wiki/English_language`
[83] `http://dbpedia.org/`
[84] `http://dbpedia.org/page/Buginese_language`
[85] `http://en.wikipedia.org/wiki/Buginese_language`
[86] `http://www.ethnologue.com/show_language.asp?code=bug`
[87] `http://en.wikipedia.org/wiki/NoSQL`

would permanently visible from the Internet), I decided to use MySQL[88], which is available on my web hosting.

3.4.1 Tables

There are two tables `w2c_alias` and `w2c_language`. The table `w2c_alias` contains two columns `alias` and `iso`. The purpose of this table is to make all scripts more user-friendly. Internally, ISO 639-3 codes are used but for the user, it is much easier to write 'czech', 'cs' instead of 'ces'.

The problem was, that some language names are the same as ISO codes of other languages. For example the En language[89] has ISO 639-3 code enc but its name is the same as the ISO 639-1 code for English. It would be very confusing if 'en' was used, and it would not mean English. For these reasons, aliases are filled in this order: ISO 639-3 codes, language name, name used on Wikipedia and local name. Now, when 'en' is used, English is used.

The second table `w2c_language` contains 3 columns `language`, `key`, `value`, where `language` represents ISO 639-3 code and `key` and `value` are arbitrary strings up to 30 characters for key and 255 characters for value.

3.4.2 Access

There are three ways how to access stored data - using web interface, simplified RESTful API[90] and script `webAPI.sh` .

The web interface is available on `http://w2c.martin.majlis.cz/language/`. It is possible to specify the language and key and all corresponding values are returned. It is possible to specify output format which can be:

- TXT - text output - columns are separated by tabs. This output may be easily processed with unix command-line tools.
- XML - XML output
- JSON - JSON[91] output which can be easily used in programs.

[88] `http://www.mysql.com/`
[89] `http://www.ethnologue.com/show_language.asp?code=enc`
[90] `http://en.wikipedia.org/wiki/REST`
[91] `http://en.wikipedia.org/wiki/JSON`

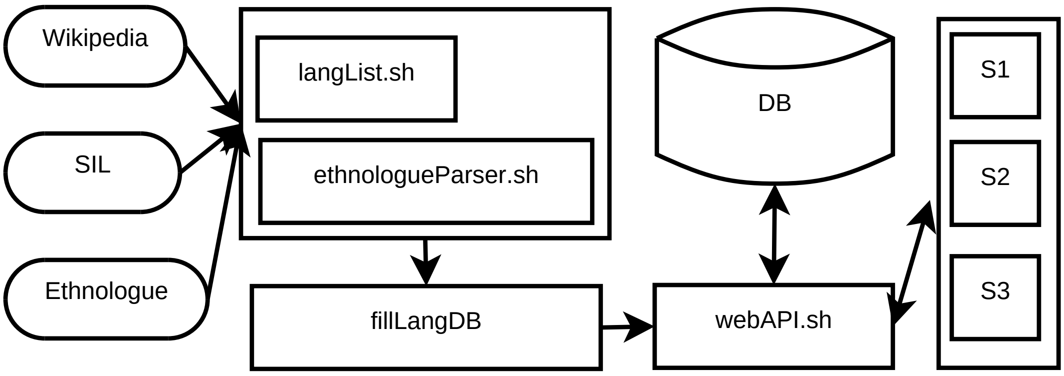


Figure 3.2: Metadata — work flow

The URLs provided by the web interface are also a part of the REST API. If proper authentication token is used, values may be changed or new ones added.

The script `webAPI.sh` is a wrapper written in bash. It uses REST API and its text output. This script is used by almost all programs.

3.4.3 Work Flow

The metadata flow is displayed in Figure 3.2. This section just connects information presented in Sections 3.3 and 3.4.

Metadata is automatically retrieved from the Internet with scripts `langList.sh` and `ethnologueParser.sh`. Downloaded information is stored in temporary text files. These files are then processed with scripts in a `fillLangDB` directory. These scripts use `webAPI.sh` for inserting data into a database. When any script (`S1`, `S2` etc.) needs any information, it uses `webAPI.sh`. Some scripts are also adding new metadata, therefore an arrow exists between scripts and `webAPI.sh` is bidirectional.

Using this metadata, it is very easy to create simple scripts. Script for building corpora from Wikipedia in languages, that do not use the latin script is showed in Example 1.

3.5 Wiki Corpus

The next step in building a web corpus was to construct the initial corpus. I decided to use Wikipedia (2.2.3), because it was widely used in other multilingual corpora and also, I have previously worked with Wikipedia. I constructed several

Example 1 Wiki corpora for languages not using latin script

```
for l in `webAPI.sh GET null script | grep -v 'Lat' | cut -f1`; do
        url=`webAPI.sh GET $l 'wiki url' | cut -f3`;
        if [ ! -z $url ]; then
                wikiCorpora.sh -c 100 $l;
        fi;
done;
```

tools, constructed several initial corpora and developed a work flow for building additional wiki corpora.

3.5.1 Tools

At the beginning I used a script `wikiMiniCorpora.sh` for downloading Wikipedia pages. It is a wrapper for `crawlerSimple.sh`. It is possible to specify, how many web pages should be downloaded from Wikipedia, and the script then downloads them. Pages with a colon or a number in their title are skipped. Pages with a colon are typically special pages (Talk:*, Wikipedia*, Special:*, User:) and pages with a number are very often 'Date pages'[92].

When I downloaded a few Wikipedias, I found out that this approach is insufficient for at least two reasons. Firstly, there were many similar sentences (automatically generated - e.g.: "This article needs additional citations for verification. Please help improve this article by adding reliable references."[93]. Secondly, it was quite slow, because the crawler has to wait for a few seconds after each request and this behaviour dramatically increased the total execution time.

The first problem was solved by the script `cleanFile.sh`, which is described in Section 3.10. To overcome the second one I developed the `wikiCorpora.sh` script.

Script `wikiCorpora.sh` downloads directly the Wikipedia dumps (provided by Wikimedia). On the one hand, it significantly improved processing speed, but on the other hand, it brought problems with parsing Wikipedia special syntax. I used the CPAN module Text::MediawikiFormat[94] to convert the wiki format to HTML

[92] e. g.: `http://en.wikipedia.org/wiki/1918`

[93] `http://www.google.com/search?q=site\%3Aen.wikipedia.org+"This+article+needs+additional+citations+for+verification."`

[94] `http://search.cpan.org/~dprice/Text-MediawikiFormat-0.05/`

and then to plain text. I found out that this module did not work correctly, so I used slightly different approach. At the beginning all links, tables and special syntax are removed. This preprocessed text is passed to the Text::MediawikiFormat module to create a HTML output, from which only paragraphs are preserved and all tags are removed. Then, duplicates lines are removed with the script `cleanFile.sh`.

3.5.2 Corpora

For prototyping, I used a corpus build from 5,500 articles for each language with at least 100 thousand articles. Later on, I extended this corpus to languages with at least 5 thousands articles. This corpus contains 115 languages. This corpus has a database key `data wiki_5500`[95].

For my main work, I used a corpus of 20,000 articles from Wikipedias with at least 5 thousands articles. This corpus has a database key `data wiki_20000`[96].

Both corpora are available as plain text files, vertical files and frequency files of trigrams and quadgrams.

3.5.3 Work Flow

The work flow for building the Wiki Corpus is displayed in Figure 3.3. The first step is script `download-wikipedias.sh` execution with a specified number of required pages and minimal article counts. This script executes `wikiCorpora.sh` for each language and created sub-corpora are stored on the disk.

It is possible to extend this process by executing script `processFiles.sh`, which iterates over languages included in the downloaded corpus. For each language, is a script `processFile.sh` executed. This script removes duplicity with `cleanFile` and generates a vertical file using `verticalFile.sh`. Frequency lists for n-grams are constructed with `frequencyList.sh`. All created files are uploaded to the hosting and URLs of these files are added to the database.

[95] `http://w2c.martin.majlis.cz/language/?lang=\&key=data+wiki\_5500*\&format=TXT`

[96] `http://w2c.martin.majlis.cz/language/?lang=\&key=data+wiki\_20000*\&format=TXT`

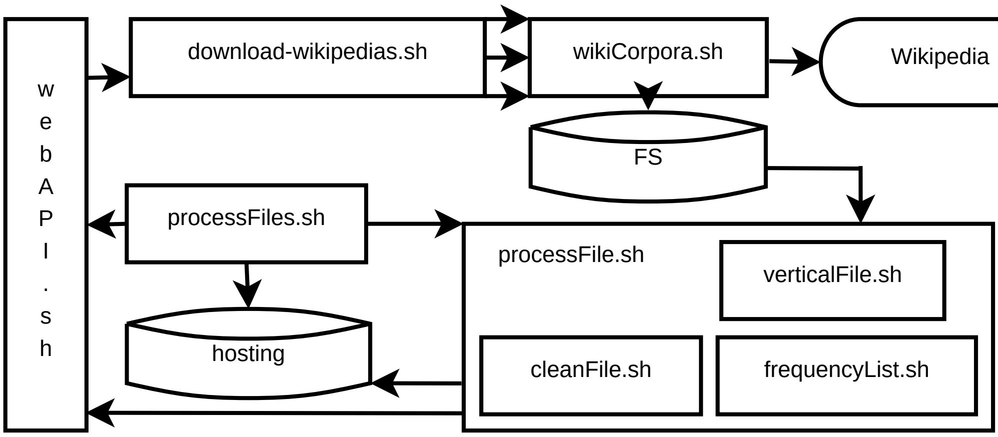


Figure 3.3: Wiki Corpora — work flow

3.6 Language Recognition

The language recognition is one of the crucial components of the project. Existing solutions, described in Section 2.7, are usually able to recognize around 10 languages. To achieve the goal, my language detector must be capable of recognizing more than ten times more languages.

3.6.1 Prototype

I started language detection with simple prototyping. I built a Wikipedia corpus for languages with at least 100 thousand articles (31 at that time) and I used two thousand of them. I used the simplest method - character n-gram model. I trained it on full sentences without segmentation or any preprocessing. For example 'I am' would create 3-grams: ‘__I’, ‘_I ’, ‘I a’, ‘ am’, ‘am_’ and ‘m__’. I trained this model for n-grams for n from 1 to 5 and I selected n-grams from the top of the frequency list until p percent of the total n-gram count was chosen. This means that for frequency list of unigrams: ‘a’: 5, ‘b’ 2, ‘c’: 1 and p equals 0.5, only ‘a’ would be chosen. Achieved results are shown in Table 3.1. It seemed that anything more than 4-grams would provide sufficient results and I considered this problem as solved.

3.6.2 Full Scale

In the next step, I ran this experiment in full scale with more than one hundred languages, and I found out that accuracy dropped significantly. The reason was that for every major language, there is set of related languages. For English,

Ratio	1-gram	2-gram	3-gram	4-gram	5-gram
0.05	0.021	0.403	0.891	0.992	0.999
0.10	0.022	0.623	0.969	0.999	0.999
0.15	0.037	0.790	0.989	0.999	0.999
0.20	0.117	0.880	0.992	0.999	0.999
0.25	0.222	0.918	0.992	0.999	0.999
0.30	0.285	0.907	0.993	0.999	0.999
0.35	0.350	0.930	0.993	0.999	0.999
0.40	0.219	0.903	0.993	0.999	0.999

Table 3.1: Language recognition for the first 31 languages

it was Welsh, Irish, Scottish Gaelic, Scots, etc. For Spanish it was Portuguese, Occitan, Catalan, Asturian, Galician, etc. For Russian, it was Bulgarian and Ukraine. The hardest was Croatian, Serbo-Croatian and Bosnian.

For example, the word 'goat' is in Occitan, Catalan, Spanish and Portuguese written as 'cabra', and in Latin, Italian and Romanian as 'capra'. Word 'bridge' is written as 'pont' in Occitan, Catalan and French, and as 'ponte' in Latin, Italian and Portuguese.

The full scale experiment used 20 thousands articles from Wikipedias with at least 5 thousand articles. One half was used for training, one third was used as heldout and the rest for testing.

The main problem was, that some minor language was often reported instead of its associated major language. I tried different ratios - top X n-ngrams, where X is either fixed or a percentage or until they covered some ratio of n-grams (the same as in the initial experiment). I also tried using segmented sentences or even lower-cased sentences. I tried using heldout data to normalize the score by score achieved per kilobyte. Some combinations of the methods mentioned above preferred the major languages and some of them preferred the minor ones. To fix this, I boosted 'the right ones' manually.

Because running the full-scale experiment took more than a day, I used an iterative approach. From the last known result, I picked a problematic pair — i.e. English and Scots — selected a method and tuned parameters to achieve good results. Then I added a few more languages from other families — Spanish, Catalan, Russian and Bulgarian - and re-ran the experiment. Very often this ended with satisfactory results, so I ran the experiment with full data. However, it very often ended with in failure — for example, most of the Occitan was recognized as Spanish. So I added Occitan to the small scale experiment and with minor

	Model 1		Model 2	
Language	Accuracy	Mistakes	Accuracy	Mistakes
Major	0.8	Min 1: 0.1, Min 2: 0.1	0.99	Min 1: 0.01
Min 1	1		0.7	Maj: 0.3
Min 2	0.9	Major: 0.1	0.7	Maj: 0.3

Table 3.2: Language recognition — model selection

tuning I tried to fix it. When it was fixed, I ran the experiment in full-scale and I found out, that there was trouble with another language.

For example, when the top 5% of 4-grams or more than 2000 4-grams were chosen, then all Russian texts were recognized as Bulgarian (all Bulgarian was recognized as Bulgarian). When I decreased the number of 4-grams to 200, only 4% of Russian texts were recognized as Bulgarian (Bulgarian was still Bulgarian). When I decreased the number of 4-grams to 100, all samples were recognized perfectly.

Decreasing the amount of n-grams dramatically increased the performance of the recognizer but few languages were still recognized instead of major languages, so I manually boosted English (eng), French (fra) and Russian (rus) by 10%.

3.6.3 Language Model Selection

Selecting the language model was the only step, when manual intervention was needed. A typical situation is displayed in Table 3.2, where 2 models are compared. Language *Major* represents a major language and languages *Min 1* and *Min 2* represent minor related languages. The first one has higher mean and median, so it looks better. If the first model will be chosen, then during the harvesting of language *Min 1*, a lot of pages in the *Major* language will be discovered and 10% of them would be recognized as *Min 1*. Due to the expected differences in many orders between languages *Major* and *Min 1*, a corpus of *Min 1* will be created from texts in *Major*. The second model, on the other hand, will throw away 30% of pages written in languages *Min 1* and *Min 2*, but the resulting corpous will still be be cleaner than the one from the first model. That is why I preferred the second model .

3.6.4 Final Version

The final version of my language recognizer was constructed in the following way. The Wiki Corpora was divided into two parts for training and testing. The first five sixths were used for training and the remaining data was used for testing. Test data for each language was divided into 500 equally large (in words) chunks. If a chunk was greater than 500 words then extra words were deleted.

Preprocessing

All input texts — for training and testing, were processed in the following way:

- All punctuation characters were removed. I only used characters class `[:punct:]`[97], because adding any Unicode punctuation character used in non-latin languages significantly increased the processing time[98]
- All digits were removed. I was using character class `[:digit:]`. When a whole word with digits was deleted, almost everything was deleted in languages without spaces - e.g. Japanese (jpn) or Chinese (zho).
- Input text was divided into words. Words are separated by character class `[:space:]`.

For example, the input sentence 'A, b568c de.' is segmented into words 'A', 'b', 'c' and 'de'. All words are separately divided into 4-grams with padding - e.g. 'I' constructs four 4-grams '___I', '__I_', '_I__' and 'I___'.

Training

The probability of each 4-gram is computed using the training data and only the first 100 are preserved. These probabilities are normalized to sum up to 1. Probabilities for English (eng), French (fra) and Russian (rus) are boosted by 10%, so they sum to 1.1. All these probabilities are treated as a score and merged into single model.

[97] `http://www.gnu.org/software/grep/manual/grep.html`

[98] Grep with [:punct:] executed on a 20MB file took 2 seconds, when character '•' was added processing took 10 minutes.

(a) Training data

Lang	Training data
L1	`bbbeaccdcdaabbbbeddc`
L2	`bbacceeceaedcdeabbeb`

(b) Training Probabilities

Lang	a	b	c	d	e
L1	0.15	0.35	0.20	0.20	0.10
L1	0.15	0.25	0.20	0.10	0.30

(c) Language Model

Uni	Lang	Score	Uni	Lang	Score
b	L1	0.43	c	L2	0.27
b	L2	0.33	d	L1	0.29
c	L1	0.29	e	L2	0.40

(d) Detection — 'aabbecdec'

Lang	Computation	Score
L1	0.00 + 0.00 + 0.43 + 0.43 + 0.00 + 0.29 + 0.29 + 0.00 + 0.29	1.73
L1	0.00 + 0.00 + 0.33 + 0.33 + 0.40 + 0.27 + 0.00 + 0.40 + 0.27	**2.00**

Table 3.3: Language recognition — example

Detection

During detection, the input text is preprocessed and divided into 4-grams. Scores for each language are summed up and the language with the highest score is the winner.

Example

A simple example for two languages, an unigrams language model and only the first 3 unigrams are used, is shown in Table 3.3. Training data (a) is used to compute probabilities (b). Only the first 3 most probable unigrams for each language are preserved, normalized and stored in the language model (c). Language detection for sample input string is presented in Table (d), so the input string 'aabbecdec' would be recognized as L2.

3.7 URL Seeds

At the beginning I used external links from Wikipedia. These external links are stored as a SQL dumps provided by Wikimedia. For retrieving these links I was using script `wikiExternalLinks.sh`. I found out, that the vast majority of these links can not be used, because pages did not no-longer exist, it were specialized websites or databases, were written in English, etc.

So I decided to use Google Search. When the user agent in the HTTP request header contained word ‘bot’, then Google returned HTTP Status Code 403 Forbidden. So I used user agents used by web browser.

I used trigram frequency file from the Wiki Corpora to generate search phrases. All trigrams with numbers or punctuation were removed and from the remaining list trigrams on lines from 2nd to 5th percentile were chosen. I used 30 queries to Google and stored the first hundred of links.

3.8 W2C Builder

The W2C Builder is a distributed corpus builder capable of running on multiple machines. For building the web corpus, several components are needed:

- crawler — receives an URL and returns HTML code
- parser — receives HTML code and return text
- detector — receives text and returns language code
- master — coordinates work of all components mentioned above

The initial plan anticipated that there will be multiple masters running in the computer laboratory, that will be coordinating all workers. But there are a few aspects, that should be considered. Not all workers use the same resources - parsing requires CPU, language detection requires CPU and memory for storing language model. It is a waste of resources to transfer data over network, when it should be completely processed on a single computer. Hence, I decided to change my plans, and instead of a single master for the whole laboratory, a master was executed on every machine. Support scripts were used for master execution and storing of total results. Even though the W2C Builder is capable of running on multiple machines, it was never really used this way, because it would not provides any benefit.

3.8.1 Overview

The W2C Builder consists of several bash and Perl scripts that cooperate with each other. Scripts `crawler.pl`, `parser.pl` and `detector.pl` are workers re-

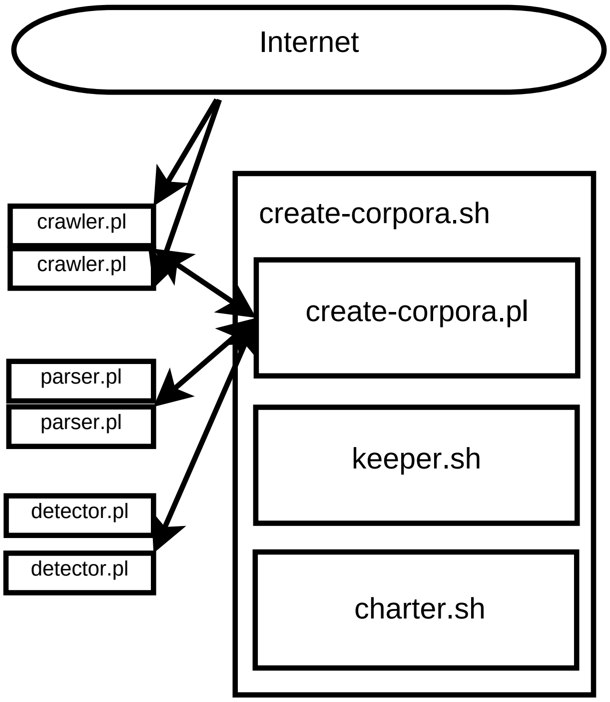


Figure 3.4: W2C Builder

sponsible for crawling, text extraction and language recognition. Script `create-corpora.pl` is the master, scripts `keeper.sh` and `charter.sh` are responsible for restarting workers and drawing charts. The `create-corpora.sh` is only a wrapper, that is executed by the user. The scripts are displayed in Figure 3.4.

For building a web corpus with 10 million words in Czech, it is sufficient to execute `./create-corpora.sh ces 10M`.

3.8.2 Configuration

The system configuration is stored in an XML file in which it is specified, how many workers are to be executed. For example, in the configuration file in Example 2, it is specified, that the master runs on the localhost and listens on port 9001. One crawler, parser and detector are to be executed at the localhost. Each task will contain 40 URLs. All workers will be executed with nice 19.

3.8.3 create-corpora.sh

The script `create-corpora.sh` is the main script executed by the end-user. For example, the command `create-corpora.sh ces 10M` creates a corpus with 10 million words for Czech. This script is responsible for argument checking —

Example 2 W2C Builder — configuration file

```
<config>
        <master logging="INFO">
                <host>localhost</host>
                <port>9001</port>
        </master>
        <crawlers logging="INFO">
                <node>localhost</node>
        </crawlers>
        <parsers logging="INFO">
                <node>localhost</node>
        </parsers>
        <detectors logging="INFO">
                <node>localhost</node>
        </detectors>
        <tmpDir>/tmp/corpora-tmp/</tmpDir>
        <resultDir>/tmp/corpora-res/</resultDir>
        <workerDir>/tmp/corpora-workers/</workerDir>
        <packSize>40</packSize>
        <commandPrefix>nice -n 19 </commandPrefix>
</config>
```

whether specified language code is available in the language recognizer. When the correct language is used, then the language model and corresponding trigram frequency list is downloaded from the hosting. The URL seed (3.7) is constructed from the downloaded frequency lists. Then, scripts `keeper.sh` and `charter.sh` are executed in the background. Then the master `create-corpora.pl` is executed. When the master finishes `keeper.sh` and `charter.sh` are killed and the downloaded results are packed with script `packData.sh`.

3.8.4 create-corpora.pl

The script `create-corpora.pl` is the master script for the W2C Builder and works as a server for all workers.

During the initialization phase, the script reads the configuration file, inserts an initial URL seed into database and builds a distribution archive. The path to the configuration file and the file with the initial URLs are passed as an argument. The distribution archive is a gzipped tar archive with source codes necessary for worker execution.

All URLs are stored in the SQLite database[99]. I decided to use this database, because it is widely available on all systems, and therefore it does not increase requirements. I wouldhave preferred to use any NoSQL database, but I did not find any widely available one without the need of additional configuration. The same problem was with traditional databases like MySQL[100] or PostgreSQL[101].

Then, distribution archives are copied on nodes specified in the configuration file and the corresponding workers are executed.

Logging is important for debugging and run analysis of complex programs, so I decided to use log4perl[102] which is compatible with log4j[103]. Apache Log4 is widely used in applications written in Java, but there are also ports for other languages. The main advantage of the widely used format is the availability of tools for log analysis[104].

[99] http://www.sqlite.org/
[100] http://www.mysql.com/
[101] http://www.postgresql.org/
[102] http://mschilli.github.com/log4perl/
[103] http://logging.apache.org/log4j/1.2/
[104] http://en.wikipedia.org/wiki/Log4j#Log_Viewers

Tasks

The task is a small unit of work, which is assigned to a waiting worker. The task is in the form of gzipped tar archives, designed in such a way, that the output from the preceding worker in the processing pipeline is the input for the following worker. The main file in the archive is called a protocol, columns are called attributes. Each row contains information about a processed URL.

The crawl task contains only a protocol with URLs. URLs are read from the database. When an URL is chosen, it is marked as 'in progress'. The crawl downloads URLs and fills attributes actual time, URLs md5 hash, HTTP Status code, base URL, charset and size. Downloaded files are added to the archive in the form of `urls-md5.html`.

The parser task is the crawler's output archive. It reads the protocol and searches for URLs with the correct attributes (HTTP status, mime-type). If a correct URL is found, the stored HTML file is processed. Links are stored in the file `urls-md5.links`, text is saved to the file `urls-md5.txt` and attributes for number of links, text size in characters and text size in words are filled in.

The detector task is the parsers's output archive. It reads the protocol and searches for URLs with the correct attributes (text size, number of links). If a correct URL is found, a language is recognized and stored to the protocol.

When the server retrieves a result from any detector, it reads the protocol and searches for URLs in the target language. If a URL is found, all links are added to the database and the text is appended to the corpus. The attributes of all URLs are stored in the database and the URL itself is marked as finished.

When a new URL is added to the database, it gets assigned a random number. When URLs are selected for a new crawler task, then the first N according to this random number are chosen. The purpose of this is to reduce the probability of all the URLs in the task being from the same domain.

This design allows reprocessing finished tasks. If the text extraction or the language detection are improved, then all finished tasks could be used as input for the parser or detector.

URL Preprocessing and Filtering

All URLs are normalized[105] to reduce the obvious duplicity on the URL level; for example, these URLs are equal `HTTP://www.Example.com/` and `http://www.example.com/`.

The URL filtering was essential for increasing the yield of the crawling. In the early versions, I started with manually written regular expressions for the most common file types (doc, docx, xls, xlsx, etc.), which should be ignored. After a few experiments, I found out, that this is not sufficient, because lot of links directed to advertisement websites. I thus decided to use a list of known advertising websites[106] as blacklist. However, further investigation revealed that there are also links to bookmarking services (digg, stumble, etc.) or social services (twitter, facebook), which should also be ignored, so I abandoned this idea.

Also, the top-level domain names can be used for filtering. When the task is to build a Czech corpus, all pages under TLD '.cz' are good candidates (Czech is used in the Czech Republic with the TLD '.cz') but pages under '.de' (Germany) are not good candidates. It would be feasible to create such rules for a few major languages, but not for hundreds. Furthermore, domains under the 'right' TLD are not always worth crawling - for example search results, catalogues, advertisement servers etc.

To solve this problem, I used an additional database with two tables - one for TLDs and one for domains. These tables contain column for the TLD (or domain name), the number of downloaded URLs, the number of valid URLs, the ratio of valid URL (in percent) and information, whether this domain is ignored.

When a URL was processed, then its TLD and domain name was extracted. The number of downloads for this TLD and domain was increased. If the URL was in the target language, than the number of valid URLs was also increased. Then, the ratios were updated. If the TLD was downloaded more than 20 times and has less then 10% of valid URLs, then it was marked as ignored. Same approach was used for domains, but at least 40 downloads were required. The ratio 10% looks very low (should be higher), but I found out, that when this ratio was higher, lot of domains were banned too quickly. Complex websites contain lot of sections with categories, tags, archives, list of articles by date, author, etc. Typical situation was, that the page with connected text was downloaded first, but lot of links

[105] http://en.wikipedia.org/wiki/URL_normalization

[106] `https://easylist.adblockplus.org/en/`

from this page links to pages with lists of articles (tages, sections, etc.) without connected text. So this domain got immediately marked for ignoring.

When whole task was processed, domains newly marked as ignored were used to mark all unprocessed URLs in database as invalid (and therefore it will not be chosen). Before any URL was added to the database, it was checked, whether it is from ignored TLD or domain.

This filtering speeds up processing twice.

3.8.5 crawler.pl

The script `crawler.pl` is responsible for downloading web pages. I used CPAN package LWPx::ParanoidAgent for downloading web pages. Downloading of URL consist of several steps. The HTTP Header is read and HTTP Status code and mime-type are extracted. Only pages with mime-type text/html and status code 2XX are processed further. In the next step, the content charset is retrieved. A complete webpage is converted to utf-8 encoding with package Text::Iconv. If conversion fails or empty content is returned, then processing of this URL is stopped. The converted webpage is normalized by tidy[107] with options `-utf8 -asxml -b -q`.

3.8.6 parser.pl

The script `parser.pl` is used for extracting texts and links from web pages. I used CPAN module HTML::Parser for parsing. The parser extracts only texts of paragraph (inside elements `<p>`). The text from the paragraph is added to the result if it is considered as valid. A valid paragraph:

- contains at least 8 words - ommits poorly written lists and headers: `<p>Item 1</p><p>Item 2</p><Item 3</p>`.
- contains less than twice more words than links - ommits menus `<p><a>Menu 1</a><br><a><Menu 2</a></p>`.
- Does not contains too much punctuation (less than 66% of words).

All these constants were empirically selected during initial phases of development.

[107] `http://tidy.sourceforge.net/`

During testing, I found out that the amount of poorly written web pages is much higher, than I expected. Therefore, usually only a very small amount of text was selected. This was caused by using `div` tags instead of `p` or by dividing long texts just by `br` tags. When the extracted text was smaller than 20% of complete webpage size, then all `div` and `td` tags were treated as `p`.

3.8.7 detector.pl

The script `detector.pl` is responsible for the language detection of downloaded texts. At the beginning, it receives the language model from the master. Only texts with at least 50 words (or 300 characters) are recognized. Language recognition is described in Section 3.6.

3.8.8 controller.pl

Program `controller.pl` is used for controlling and monitoring. The main commands are:

- `nodes` — returns a list of nodes along with their statistics
- `status` — returns information about progress
- `terminate` — terminates `create-corpora.pl`
- `addCrawler` — executes a new crawler on specified computer
- `addParser` — executes a new parser on specified computer

3.8.9 keeper.sh

The script `keeper.sh` is crucial for keeping the W2C Builder running. If fewer workers, than specified in the configuration file are running, then it executes missing workers. If the job queue is empty, then it terminates `create-corpora.pl`. This script also removes zombie processes.

3.8.10 charter.sh

This script `charter.sh` is responsible for collecting statistics, e.g. queue sizes, the amount of downloaded URLs, the amount of processed texts, the number of running workers, etc. These statistics are stored in a file (each column has specific meaning). This file is parsed and different charts are generated.

3.8.11 create-corpora-local.sh

The script `create-corpora-local.sh` is simply a wrapper for textttcreate-corpora.sh. I found out, that in the computer laboratory, it is not possible to connect to the localhost and that the hostname has to be specified. Therefore, this script creates a new configuration file `config-HOSTNAME.xml` from the main configuration file and replaces all strings 'localhost' with the actual hostname. The path to this configuration file is used as an argument for `create-corpora.sh`.

3.9 Distributed Corpus Building

To maximize the computer laboratory utilization, a distributed approach was applied. In the early stages, I was executing jobs manually; typical work flow looked like this:

- Find a node, where the downloading finished or crashed.
- Connect to that node.
- Copy the result to the ufallab (type password (3.1)).
- Delete results and temporary files.
- Execute new job.

The problem with this approach was, that it required constant supervision, because every 15 minutes, some interaction was needed. I realized that this method was not efficient in terms of human work, so I started to work on a fully automatic process. On the server ufallab runs webserver Apache[108] with the option `post_max_size`[109] set to 8MB. I used 7-zip[110] to compress the result and to create 4MB volumes. Then this volumes were encoded to base64[111] encoding and uploaded via HTTP Post method to the ufallab. On the ufallab was simple PHP script, that stored received volumes on the disk and created a bash script for their

[108] `http://httpd.apache.org/`
[109] `http://php.net/manual/en/ini.core.php\#ini.sect.data-handling`
[110] `http://www.7-zip.org/`
[111] `http://en.wikipedia.org/wiki/Base64`

processing. This included extraction, decoding and moving to the folder with results deletion of received files. This solution did not require human interaction, but was very inefficient. The amount of data, that was downloaded during six hours in the computer laboratory took almost one day to compress and upload and another day to merge and decompress.

My final solution was a semi-automatic process, that required typing in the password only a few times a day. It consists of two scripts — `fill-corpora-quotas.sh` and `copy-results-to-single-node.sh`.

3.9.1 fill-corpora-quotas.sh

The purpose of the script `fill-corpora-quotas.sh` is to build a corpus of specified size for all languages passed as an argument. It processes languages sequentially and reads information about the actual corpus size[112] from which it retrieves the amount of data downloaded for the language. The size of jobs already running for the language is added. If the total size is smaller than the quota, then a free node is located and the W2C Builder for the language is executed. If a free node is found and the builder executed, the size of already running jobs is increased.

For example, if there is 85MW downloaded for language X and the limit is 100MW, only 2 jobs for 20MW will be executed.

3.9.2 fill-corpora-quotas-wrapper.sh

The script `fill-corpora-quotas-wrapper.sh` executes the script `fill-corpora-quotas.sh` for each language.

3.9.3 copy-results-to-single-node.sh

The script `copy-results-to-single-node.sh` copies all downloaded data to a single computer in the computer laboratory from which it can be manually uploaded to ufallab.

[112] `http://ufallab.ms.mff.cuni.cz/\~majlis/info.txt` — generated by `generate-stats.sh`

3.9.4 merge-results.sh

The script `merge-results.sh` is executed on the server ufallab. This scripts takes all partial results and merges them into a single corpus. Each language is treated separately, so when new results for a single language are added, only data related to this language are processed.

In the early versions, when texts were merged, duplicate content was removed. But when the collected data grew, it took almost three days to merge the results, that were downloaded in the computer laboratory during a single day. It would have not been feasible to distribute the merging to the computers in the computer laboratory, because the bottleneck was the single disk on ufallab.

3.9.5 generate-stats.sh

The script `generate-stats.sh` is responsible for generating statistics about downloaded data. The statistics are generated in machine readable form[113] for other scripts and in human readable form[114].

3.10 Duplicity Detection

When I was processing Wikipedia, I found out, that many pages contained the same paragraphs, even though they were not duplicates. It would be wasteful to throw away the complete web page, especially for minor languages. So instead of page oriented approach, I decided to use paragraph oriented approach. I created frequency list of all paragraphs and the first 1 percentile of lines were removed. It removed the most duplicate lines. I also removed the last 1 percentile of lines, because they contained malformed lines (“, , , , , , ,”, “)))”, etc.). The remaining lines were shuffled. This was done by `cleanFile.sh` script.

Then I realized that removing duplicities on the line level could also solve the problem with spam pages and spam comments.

A good position in search engine results is now crucial for business success. There are thousands of pages trying to sell the same product, but users usually click only on the top few links. Therefore, spammers are trying to manipulate with the

[113] `http://ufallab.ms.mff.cuni.cz/\~majlis/info.txt`
[114] `http://ufallab.ms.mff.cuni.cz/\~majlis/`

search engine indexing (this technique is called spamdexing[115]). They build link farms[116] or scaper sites[117] - automatically generated websites that are tightly-knit pages referring to each other. Content is typically generated from Wikipedia or other publicly available resources. To trick the search engines, these websites do not contain exact copies of original texts, but rather only mixed fractions. These spamdexing techniques may cause problems during crawling. If breadth-first approach is used, then the crawler may get stucked in this farm. It may also fool the duplicity detection.

Another technique used by spammers, is spamming blogs[118], where bots comment blog spots. These comments contain links to the spammers' website to increase its popularity. Projects like Honey Pot[119] or Akismet[120] are catching millions of spam comments every day. Spam in comments may also be the source of duplicities and therefore decrease the corpus quality. When a blogger writes a spot on his/her blog in language X, the text is valuable for the corpus. Later, when a few spam comments are attached, this article will still be recognized as language X, but it will not be so valuable, because it will also contain some English sentences. When many such articles are added, the same comments may be presented many times.

Removing duplicate lines may also fix the problem with incorrectly recognized boilerplate code.

For these reasons I decided to remove duplicities on the line level instead of the page level.

3.11 W2C Corpus

The W2C Corpus was one of the main goals of this thesis. When I reached the quota 100 million words for many languages, I started with the cleaning of the downloaded material. For the construction of the clean corpus the following scripts were used: `process-results.sh`, `process-results-wrapper.sh` and `process-results-overview.sh`.

[115] `http://en.wikipedia.org/wiki/Spamdexing`
[116] `http://en.wikipedia.org/wiki/Link_farm`
[117] `http://en.wikipedia.org/wiki/Scraper_site`
[118] `http://en.wikipedia.org/wiki/Spam_in_blogs`
[119] `http://www.projecthoneypot.org/statistics.php`
[120] `http://akismet.com/`

3.11.1 process-results.sh

The scripts `process-results.sh` prepares the corpus for a single language. It includes of downloading the raw corpus from ufallab, computing various statistics for the Wiki Corpus, estimating the Internet, duplicity reduction and computing statistics for the W2C Corpus.

3.11.2 process-results-wrapper.sh

The scripts `process-results-wrapper.sh` executes the script `process-results.sh` for each language included in the raw web corpus.

3.11.3 process-results-overview.sh

The script `process-results-overview.sh` generates statistics about this cleaned corpus. The statistics are generated in the machine readable form[121] for other scripts and in the human readable form[122].

3.12 Corpus Distribution

At the end, is the final distribution is compiled. There are two scripts responsible for the distribution process — `build-package.sh` and `build-package-wrapper.sh`.

The W2C-97-10 Corpus was extracted from the W2C Corpus. This corpus contains 10 million words for each of 97 languages.

The unclear legal status of the downloaded material does not allow the publishing of the W2C-97-10 Corpus on the Internet, so it was released for internal usage only.

3.12.1 build-package.sh

The script `build-package.sh` downloads the complete text forhe specified language from ufallab. From this file, only a required amount of text is extracted and copied to the single computer.

[121] `http://w2c.martin.majlis.cz/w2c//data/results.eye.txt`
[122] `http://w2c.martin.majlis.cz/w2c//data/process-results-overview.html`

3.12.2 build-package-wrapper.sh

The scripts `build-package-wrapper.sh` executes the script `build-package.sh` for each language included in the raw web corpus.

3.13 Comparing Wiki vs Web

For comparing both corpora I used the following methods:

- Average Word Length
- Average Sentence Length
- Conditional Entropy — $H(Y|X) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y|x)$
- Conditional Perplexity — $G(Y|X) = 2^{H(Y|X)}$

When I collected data for the first time, I found out, that average word length for the Internet was higher than for Wikipedia. I found out, that this was really caused by the bug in the parser. When the tag `<br>` was found, it was erased and many web pages in Japanese, were instead of paragraphs using break lines. It was not possible to parse all pages again, because it would required 5 thousand hours.

I found out that achieved values depends on preprocessing. For example, for Chinese it was possible to achieve conditional entropy from 1 to almost 5. If all words with punctuation or number were removed, then the conditional entropy was 1, because all bigrams were seen just once. But when the numbers and punctuation characters were treated as a separate word, then the conditional entropy was 4.8, because many bigrams were just number with its unit.

4. Results

This chapter describes the collected results. At the beginning of this chapter, the Wiki Corpus (4.1) size is presented, followed by results for the language recognizer (4.2). Then the results for the W2C Corpus 4.3 and its comparison with the Wiki Corpus are presented. At the end of this chapter is an estimate of the size of the Internet.

Tables are sorted alphabetically according to the ISO 639-3 code. All used codes are in Table B. The highest five values in each column are printed $\overline{\textit{overlined}}$ and the lowest five are printed <u>*underlined*</u>.

4.1 Wiki Corpus

The Wiki Corpus was built from Wikipedias with at least 5000 articles corresponding to 122 languages. Methods used for building this corpora are described in Section 3.5. These 122 languages are used by 4.6 billion people (2/3 of the total population).

The complete data is presented in Table 4.1 and visualized in Figure 4.1.

The biggest outlier is the Kannada language (kan) which with just 10 thousand articles has 118MB. It seems that many articles are complete translations of articles from English Wikipedia[123]. The Kannada language is written in the Kannada script which consumes 3 bytes per character[124], so it may contains up to 3 times less characters. A similar explanation also applies for languages Thai (tha), Gujarati (guj) and Burmese (mya).

4.2 Language Recognition

The language recognition was trained and evaluated on the Wiki Corpus. Methods used for language detection are described in Section 3.6.

[123] e.g. `http://kn.wikipedia.org/wiki/\%E0\%B2\%B5\%E0\%B3\%87\%E0\%B2\%B2\%E0\%B3\%8D\%E0\%B2\%B8\%E0\%B3\%8D` — and other articles about countries

[124] `http://www.unicode.org/charts/PDF/U0C80.pdf` — Kannada Script

Lang	Size	Art	Lang	Size	Art	Lang	Size	Art	Lang	Size	Art
afr	25348	18	fas	40272	159	lat	8930	56	sah	4595	8
als	15190	10	fin	55427	273	lav	17714	35	scn	4587	17
amh	3752	11	fra	$\overline{99152}$	$\overline{1126}$	lim	6679	7	sco	5219	7
ara	57379	152	fry	16404	21	lit	24326	136	sgs	$\underline{0}$	13
arg	9890	26	gan	$\underline{920}$	$\underline{6}$	lmo	6063	21	slk	29200	126
arz	8788	7	gla	3020	8	ltz	12237	33	slv	18436	119
ast	11634	15	gle	12178	13	mal	42828	19	spa	$\overline{105891}$	$\overline{802}$
aze	27808	74	glg	30382	73	mar	10835	34	sqi	20443	33
bcl	1604	$\underline{5}$	glk	2314	$\underline{6}$	mkd	43384	47	srp	46074	145
bel	26792	38	guj	54981	21	mlg	7018	20	sun	7303	15
ben	21103	22	hat	$\underline{984}$	53	mon	11681	$\underline{6}$	swa	11282	22
bos	22804	31	hbs	28723	44	mri	1553	7	swe	39858	403
bpy	20720	25	heb	80910	121	msa	30708	121	tam	42169	34
bre	10688	38	hif	$\underline{973}$	$\underline{5}$	mya	40937	$\underline{6}$	tat	6994	12
bug	$\underline{392}$	9	hin	38859	99	nap	3299	43	tel	12291	48
bul	51853	119	hrv	36790	100	nds	19561	18	tgk	5123	9
cat	49694	346	hsb	2450	7	nep	19201	15	tgl	6491	53
ceb	3070	43	hun	50084	195	new	9788	70	tha	64411	68
ces	42328	201	hye	18132	14	nld	59434	738	tur	39112	170
chv	9074	13	ido	14132	22	nno	14352	71	ukr	77036	302
cos	1359	$\underline{6}$	ina	3208	$\underline{6}$	nor	35170	308	urd	19138	17
cym	10264	33	ind	31301	168	oci	8772	30	uzb	3779	8
dan	24431	152	isl	11370	32	oss	3324	8	vec	4543	9
deu	$\overline{136894}$	$\overline{1259}$	ita	73510	$\overline{819}$	pam	2723	7	vie	48960	211
diq	1832	11	jav	5321	35	pms	5724	41	vol	19384	119
ell	72189	63	jpn	95234	759	pnb	3750	17	war	2196	102
eng	$\overline{170366}$	$\overline{3683}$	kan	$\overline{118430}$	11	pol	49864	$\overline{815}$	wln	3174	12
epo	20495	148	kat	43485	50	por	68096	690	yid	13311	9
est	22945	86	kaz	31429	56	que	1536	17	yor	1635	30
eus	16086	103	kor	38020	167	ron	28735	163	zho	48508	365
fao	2580	$\underline{5}$	kur	8368	16	rus	$\overline{130610}$	736			

Table 4.1: Wiki Corpora — size in kB

Columns — *Lang*: ISO 639-3 code, *Size*: text size in kB after duplicity reduction, *Art*: number of articles in thousands

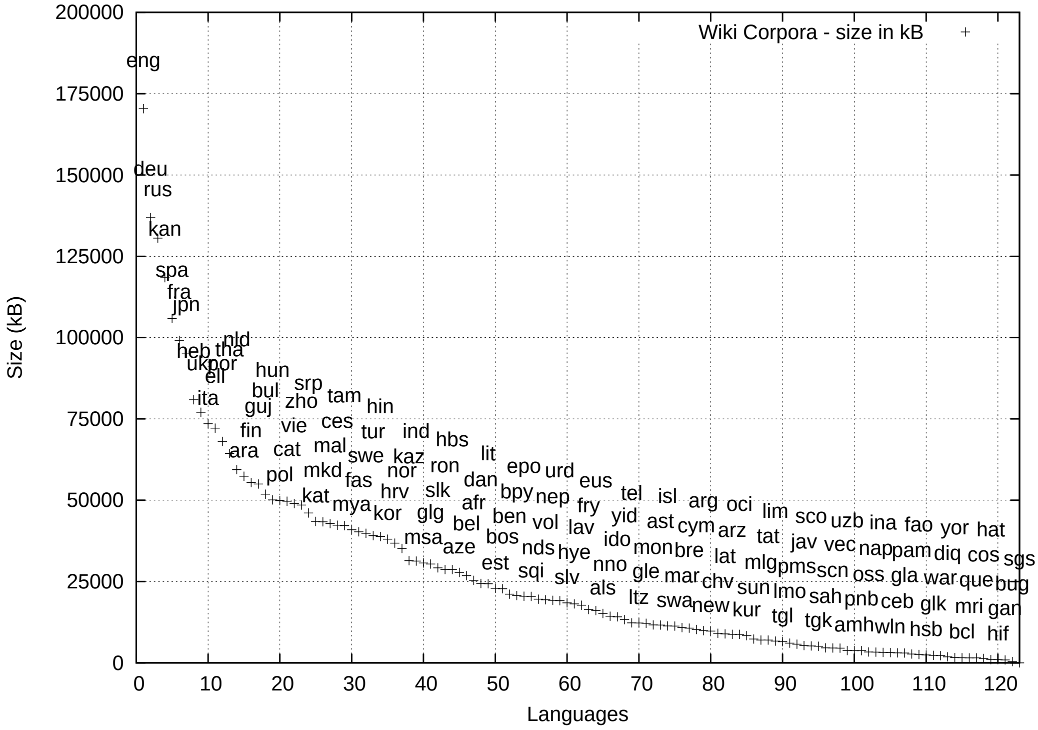


Figure 4.1: Wiki Corpora — size in kB

Languages are sorted according to their size in the Wiki Corpora.

122 languages in total were evaluated on 61 thousands test samples. The recognizer achieved in total accuracy 0.885 (with median 0.982). The detector recognized 27 languages without any errors, 52 with accuracy higher than 0.99 and 92 higher than 0.9. The histogram and the quantiles are presented in Table 4.2.

Complete results for all languages are in Table 4.3. Languages sorted according to the detector's accuracy are displayed in Figure 4.2.

There are several reasons for some of the mistakes that were made. The Wiki Corpus was constructed without any manual interaction, so not all of the training and the testing data was in a single language. This is because:

- It is a quite common practice that a new article starts out as a copy of the article from the English Wikipedia, and only after that it is slowly translated into the target language. This practice is quite common especially for minor languages[125].
- I suppose, that a similar practice could be used between close languages, but I did not find any evidence.

[125] `http://new.wikipedia.org/wiki/\%E0\%A4\%AD\%E0\%A4\%BE\%E0\%A4\%B7\%E0\%A4\%BE`

(a) Results

Languages	122
Total	61000
Success	53997
Fail	7003
1st Qu.	0.912
Median	0.982
Mean	0.885
3rd Qu.	0.998

(b) Histogram

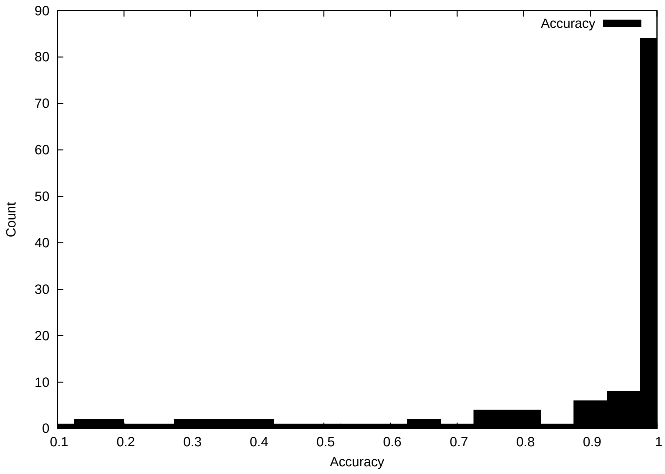


Table 4.2: Language Detection — Overview

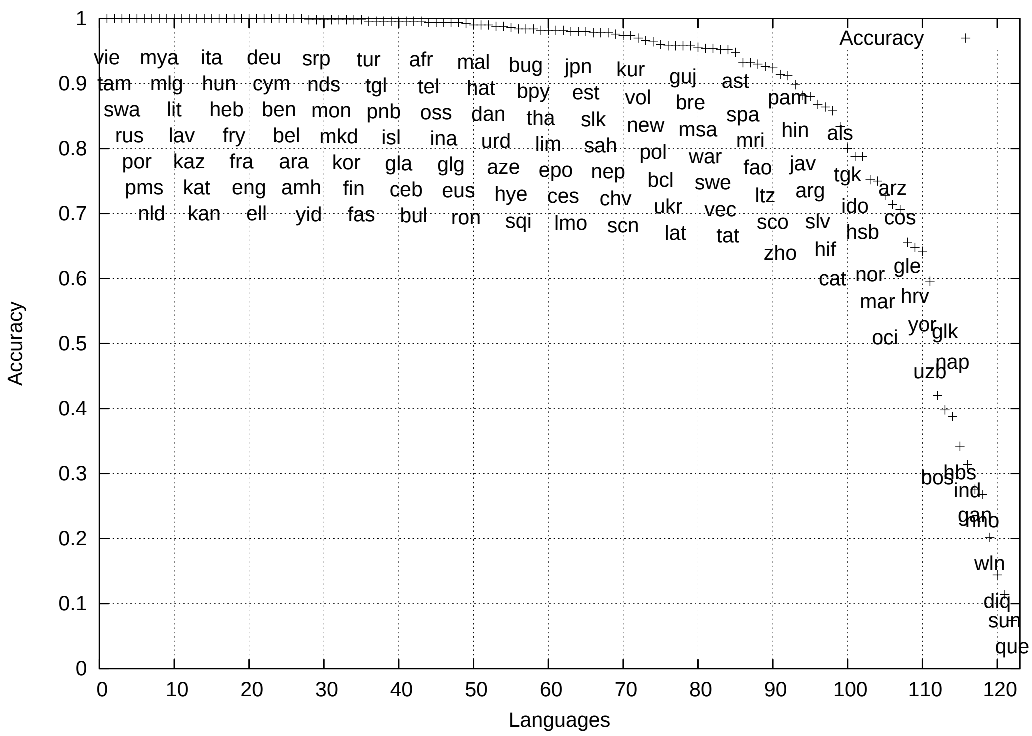


Figure 4.2: Language Detection

Corresponding table — 4.3

Lang	Acc	Sim	Lang	Acc	Sim	Lang	Acc	Sim
afr	0.996	fra: 0.002	hat	0.990	eng: 0.010	nor	0.752	dan: 0.230
als	0.834	fra: 0.050	hbs	0.342	bos: 0.342	oci	0.728	fra: 0.260
amh	1.000		heb	1.000		oss	0.994	rus: 0.006
ara	1.000		hif	0.864	swa: 0.122	pam	0.912	eng: 0.086
arg	0.880	glg: 0.092	hin	0.898	eng: 0.090	pms	1.000	
arz	0.714	ara: 0.280	hrv	0.648	hbs: 0.208	pnb	0.996	urd: 0.004
ast	0.948	fra: 0.044	hsb	0.788	swa: 0.192	pol	0.964	swa: 0.032
aze	0.988	tur: 0.010	hun	1.000		por	1.000	
bcl	0.960	tgl: 0.032	hye	0.986	eng: 0.014	que	0.074	swa: 0.890
bel	1.000		ido	0.788	ita: 0.096	ron	0.992	vec: 0.004
ben	1.000		ina	0.994	vec: 0.004	rus	1.000	
bos	0.420	hrv: 0.306	ind	0.314	msa: 0.672	sah	0.978	rus: 0.022
bpy	0.984	vie: 0.012	isl	0.996	fao: 0.004	scn	0.974	ita: 0.022
bre	0.958	eng: 0.020	ita	1.000		sco	0.924	eng: 0.076
bug	0.984	swa: 0.016	jav	0.882	pam: 0.054	slk	0.978	swa: 0.006
bul	0.996	mkd: 0.004	jpn	0.980	eng: 0.006	slv	0.868	fra: 0.028
cat	0.858	fra: 0.066	kan	1.000		spa	0.932	glg: 0.060
ceb	0.996	tgl: 0.004	kat	1.000		sqi	0.984	fra: 0.006
ces	0.982	slk: 0.010	kaz	1.000		srp	0.998	mkd: 0.002
chv	0.976	rus: 0.012	kor	0.998	vie: 0.002	sun	0.114	swa: 0.392
cos	0.706	scn: 0.236	kur	0.974	eng: 0.008	swa	1.000	
cym	1.000		lat	0.958	eng: 0.036	swe	0.954	nds: 0.040
dan	0.990	fry: 0.006	lav	1.000		tam	1.000	
deu	1.000		lim	0.982	fra: 0.012	tat	0.952	sah: 0.032
diq	0.144	swa: 0.714	lit	1.000		tel	0.994	eng: 0.006
ell	1.000		lmo	0.980	vec: 0.020	tgk	0.800	rus: 0.150
eng	1.000		ltz	0.926	fra: 0.050	tgl	0.996	pam: 0.002
epo	0.982	fra: 0.010	mal	0.990	eng: 0.010	tha	0.982	eng: 0.018
est	0.980	swa: 0.018	mar	0.750	eng: 0.242	tur	0.996	ron: 0.002
eus	0.994	fra: 0.002	mkd	0.998	swa: 0.002	ukr	0.958	rus: 0.036
fao	0.930	scn: 0.018	mlg	1.000		urd	0.988	eng: 0.012
fas	0.998	arz: 0.002	mon	0.998	sah: 0.002	uzb	0.596	tgk: 0.314
fin	0.998	eng: 0.002	mri	0.932	swa: 0.058	vec	0.952	ita: 0.032
fra	1.000		msa	0.956	eng: 0.018	vie	1.000	
fry	1.000		mya	1.000		vol	0.970	fin: 0.026
gan	0.276	zho: 0.172	nap	0.388	scn: 0.570	war	0.954	swa: 0.044
gla	0.996	eng: 0.004	nds	0.998	eng: 0.002	wln	0.202	fra: 0.794
gle	0.656	gla: 0.330	nep	0.978	eng: 0.022	yid	0.998	heb: 0.002
glg	0.994	por: 0.006	new	0.966	eng: 0.030	yor	0.642	eng: 0.134
glk	0.398	fas: 0.562	nld	1.000		zho	0.914	eng: 0.030
guj	0.958	eng: 0.042	nno	0.268	nor: 0.702			

Table 4.3: Language Detection

Columns — *Lang*: ISO 639-3 code, *Acc*: accuracy and *Sim*: the most similar language

- Some articles contain texts in multiple languages. These articles are about some wide spread texts such as famous passages from books, songs or anthems. For example, this article[126] contains a single song in eleven languages along with their transcriptions.
- I preferred the best language model for constructing a web corpus, not for achieving the highest accuracy (3.6.3).

Languages that were incorrectly recognized, have some common properties. Their training data was small, such as Buginese (bug), Dimli (diq), (gan) Gan Chinese, Gilaki (glk), Quechua (que), etc or there is a very similar language: a) Norwegian (Nynorsk) (nno) and Norwegian (Bokmål) (nor); b) Walloon (wln) and French (fra), both are Frech family[127]; Gilaki (glk) and Farsi (fas) where Gilaky was strongly influenced from Faris[128].

4.3 W2C Corpus

The W2C Corpus was the main goal of this thesis. Methods used for its construction are described in Section 3.11.

4.3.1 Execution Statistics

To build this corpus, more than 100 million web pages were downloaded and this downloading took more than 7.6 thousand hours (approximately 317 days) of computer time. The real time as well as the number of downloaded URLs is higher, because keeping track of these statistics was not done in the early stages. The number of downloaded URLs may even be higher, because when a component crashed, all actually processed web pages were lost. If time consumed for transferring data between nodes, time needed for duplicity reduction, time for computing quality metrics, and time for building distribution packages would be added, then more than one year of computer time was consumed. These statistics are summarized in Table 4.4.

The absence of statistics from the early phases caused, that some ratios, which should be in range 0–1, are higher. They were also added at various times, so the

[126] http://sk.wikipedia.org/wiki/Hej,_Slov\%C3\%A1ci
[127] http://www.ethnologue.com/show_family.asp?subid=304-16
[128] http://www.ethnologue.com/show_language.asp?code=glk

URL (k)	103935.444	URL L	64308.829	Ratio	0.618
Downloaded (MB)	4556544.624	Downloaded L	3335991.519	Ratio	0.732
Text (MB)	173785.867	Text L	131314.057	Ratio	0.755
Words (MW)	23278.877	Words L	17369.166	Ratio	0.746
Execution Time (h)	7620.656	Chunks	2231.000	Data (MB)	964755.611

Table 4.4: Web Corpora — execution statistics

URL: number of downloaded URLs, *Downloaded*: amount of downloaded data in MB, *Text*: size of extracted text in MB, *Words*: number of words; suffix *L* means in target language; *Execution Time*: consumed computer time in hours and *Chunks*: number of executed jobs

metrics that should correlate could have outliers.

4.3.2 Corpus Size

The W2C Corpus contains a total of 10.6 billion words in 97 languages, with total size being almost 90GB. The size of the collected material is presented in Table 4.5. The collected size differs for various languages, that is why 64 languages have more than 100 million words, 75 more than 50 million and 97 more than 10 million of words.

Languages with the highest amount of collected material are Malayalam (mal), Thai (tha), Japanese (jpn), Burmese (mya) and Chinese (zho). The thing all these languages have in common is that do not use space for separating words. There was a bug in script `fill-corpora-quota.sh` (3.9.1).

Table 4.6 presents the yield of crawling for different languages. Column *URL* represents the amount of unique URLs from all downloaded URLs. The average value is 0.536, but for small languages, such as Tosk Albanian (als), Haitian (hat), Gujarati (guj), etc., it is only around 0.1. Languages, that are not presented in this corpus have this ratio only around 0.02. This metrics represents, how big the Internet is for specific language. Column *Dup* represents, how much text remains after duplicity reduction.

4.4 Comparing Wiki vs Web

Comparing the Wiki Corpus and the Web Corpus is one of the possibilities how to check whether reliable data was downloaded. Several different properties may point to a language for which suspicious material was collected.

ISO	Size	Words	ISO	Size	Words	ISO	Size	Words
afr	741.140	125.684	hif	98.432	18.741	nor	963.056	153.024
als	134.228	19.956	hin	1254.703	125.470	oci	305.517	30.551
amh	202.222	20.222	hrv	818.463	119.687	pam	178.985	24.565
ara	1261.417	126.141	hun	869.870	106.172	pol	920.957	119.613
ast	136.435	21.328	hye	445.940	44.594	por	804.510	119.560
aze	633.229	68.768	ina	77.814	11.891	ron	852.413	128.423
bel	1113.988	111.398	ind	782.032	113.751	rus	1963.244	196.324
ben	1489.417	148.941	isl	777.660	110.613	sah	586.165	58.616
bos	657.471	100.145	ita	959.041	135.908	scn	253.118	25.311
bre	251.128	42.527	jav	136.911	16.139	sco	511.439	84.363
bul	1268.706	126.870	jpn	3505.917	350.591	slk	970.614	132.368
cat	732.564	119.962	kan	1727.014	172.701	slv	708.846	107.655
ces	1244.336	166.429	kat	1851.489	185.148	spa	864.982	137.491
cym	496.677	81.339	kaz	1030.728	103.072	sqi	650.159	103.415
dan	714.138	109.872	kor	1258.226	125.822	srp	1037.876	103.787
deu	770.740	107.150	kur	360.505	54.793	swa	687.184	102.848
ell	1764.323	176.432	lat	451.212	58.068	swe	837.778	128.774
eng	835.683	138.522	lav	1437.623	172.097	tam	2235.889	223.588
epo	941.651	133.936	lim	237.350	32.920	tat	426.672	42.667
est	958.051	125.339	lit	1078.219	113.359	tel	1559.062	155.906
eus	702.816	86.490	lmo	215.487	34.049	tgk	511.233	51.123
fao	159.554	22.563	ltz	451.970	72.264	tgl	641.369	101.669
fas	1153.468	115.346	mal	3614.134	361.413	tha	3582.302	358.230
fin	1215.412	129.699	mar	1379.852	137.985	tur	1033.919	119.865
fra	800.397	122.345	mkd	1194.824	119.482	ukr	1210.234	121.023
fry	656.543	98.340	mlg	93.080	13.700	urd	1164.416	116.441
gla	156.379	25.486	mon	1186.394	118.639	uzb	359.062	42.583
gle	633.214	96.461	mri	54.203	10.251	vec	112.725	18.186
glg	642.326	101.009	msa	804.109	108.043	vie	648.241	105.097
guj	1039.488	103.948	mya	3132.473	313.247	yid	1045.940	104.594
hat	114.674	21.319	nds	116.133	17.179	zho	2883.315	288.331
hbs	789.628	122.806	nep	1291.312	129.131			
heb	1124.252	112.425	nld	869.751	139.009			

Table 4.5: Web Corpora — size

Columns — *Size*: size in MB, *Words*: words in millions

ISO	URL	Dup	ISO	URL	Dup	ISO	URL	Dup	ISO	URL	Dup
afr	0.616	0.574	fry	0.474	0.568	lav	0.610	0.649	sco	0.678	0.569
als	0.080	0.616	gla	0.112	0.629	lim	0.687	0.237	slk	0.639	0.698
amh	0.382	0.247	gle	0.181	0.705	lit	0.790	0.683	slv	0.763	0.596
ara	0.778	0.773	glg	0.520	0.782	lmo	0.292	0.351	spa	0.592	0.898
ast	0.460	0.338	guj	0.106	0.869	ltz	0.295	0.482	sqi	0.531	0.889
aze	0.457	0.761	hat	0.081	0.758	mal	0.538	0.920	srp	0.434	0.746
bel	0.369	0.770	hbs	0.399	0.678	mar	0.408	0.854	swa	0.268	0.648
ben	0.630	0.791	heb	0.830	0.550	mkd	0.579	0.621	swe	0.741	0.744
bos	0.651	0.658	hif	0.109	0.949	mlg	0.113	0.373	tam	0.590	0.856
bre	0.166	0.268	hin	0.580	0.916	mon	0.440	0.869	tat	0.299	0.561
bul	0.641	0.789	hrv	0.394	0.524	mri	0.127	0.671	tel	0.496	0.919
cat	0.587	0.755	hun	0.704	0.733	msa	0.633	0.716	tgk	0.366	0.418
ces	0.740	0.564	hye	0.174	0.606	mya	0.464	0.871	tgl	0.759	0.571
cym	0.796	0.298	ina	0.822	0.096	nds	0.116	0.441	tha	0.559	0.690
dan	0.704	0.710	ind	0.392	0.696	nep	0.375	0.919	tur	0.863	0.514
deu	1.047	0.567	isl	0.626	0.915	nld	0.854	0.747	ukr	0.721	0.578
ell	0.796	0.877	ita	0.853	0.800	nor	0.604	0.531	urd	0.434	0.766
eng	1.095	0.675	jav	0.378	0.167	oci	0.319	0.558	uzb	0.148	0.873
epo	0.688	0.559	jpn	0.646	0.756	pam	0.687	0.259	vec	0.321	0.256
est	0.623	0.755	kan	0.446	0.886	pol	0.663	0.628	vie	0.779	0.560
eus	0.685	0.379	kat	0.694	0.565	por	0.737	0.575	yid	0.644	0.811
fao	0.131	0.847	kaz	0.833	0.239	ron	0.632	0.776	zho	0.580	0.903
fas	0.722	0.820	kor	0.673	0.710	rus	0.602	0.816			
fin	0.761	0.797	kur	0.171	0.706	sah	0.151	0.788			
fra	0.970	0.634	lat	0.551	0.631	scn	0.778	0.253			

Table 4.6: Web Corpus — yield

URL: Ratio between unique and all downloaded URLs; *Dup*: Ratio between text size after removing duplicate URLs and after duplicity reduction;

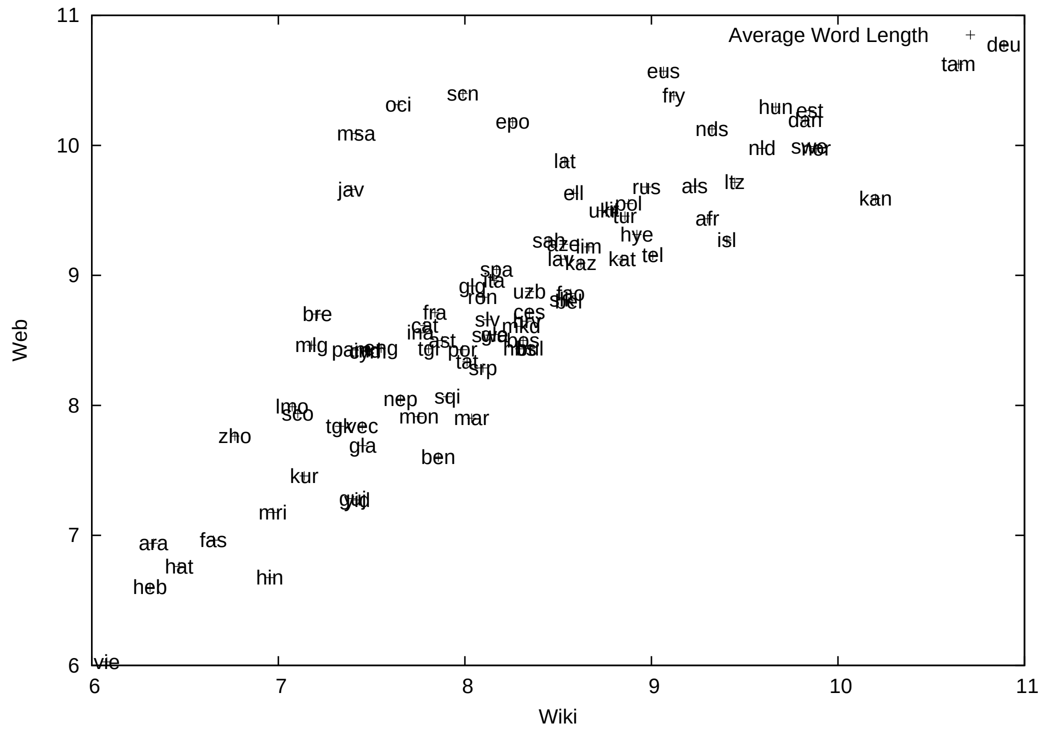


Figure 4.3: Wiki vs Web — average word length

Raw data are in Table C.1

For comparing Wikipedia and the Internet are used following properties:

- Average Word Length (4.4.1)
- Average Sentence Length (4.4.2
- Conditional Entropy and Perplexity (4.4.3)

The values presented should be used with caution, because their main purpose was only the comparison of both corpora. The numbers can be significantly changed by different preprocessing, as was shown in Section 3.13.

4.4.1 Average Word Length

The average word length may reveal problems caused by HTML parsing. The raw data are presented in Table C.1 and visualized in Figure 4.3.

The biggest outlier is Japanese (jpn), where the length differs about 66%, but it was caused by the bug.

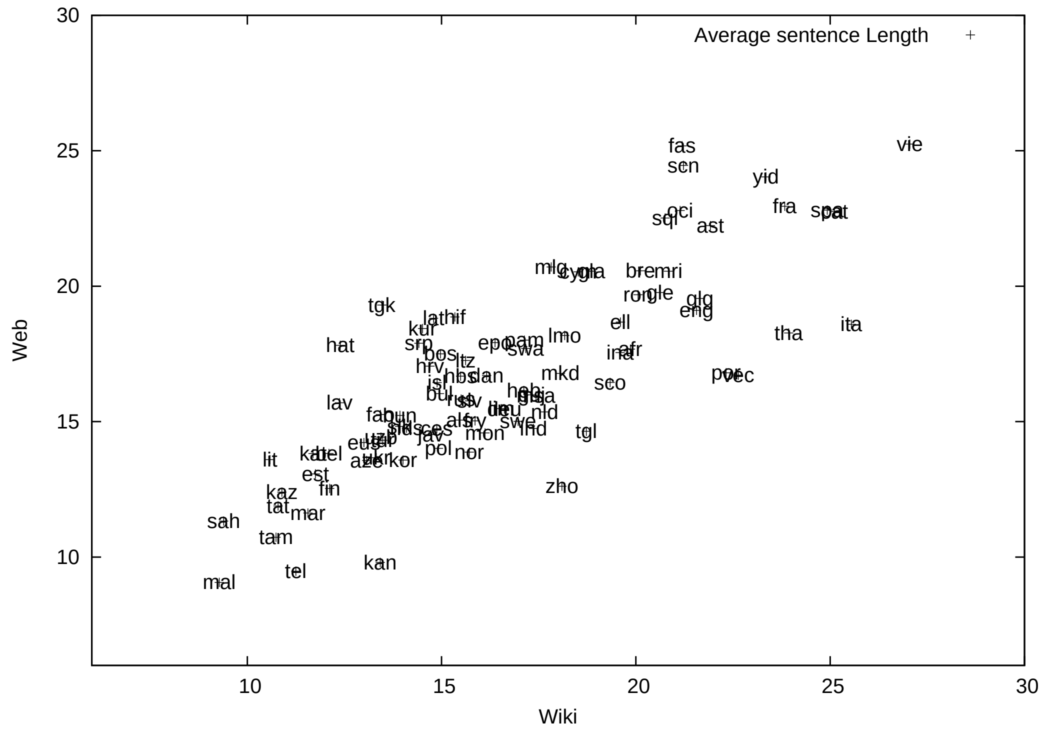


Figure 4.4: Wiki vs Web — average sentence length

Raw data are in Table C.2

4.4.2 Average Sentence Length

The average sentence length is presented in Table C.2 and visualized in Figure 4.4.

The biggest outliers in this metric are Urdu (urd) and Japanese (jpn). The average sentence length for Urdu is 429.52 in Wikipedia and only 151.94 on the Internet. Checking any page on Urdu Wikipedia[129] reveals, that it does not contain any dot, so whole paragraph is treated as a single sentence, whereas extracted segments from the Internet are much shorter and this is causing the difference. The Japanese is on the opposite site, sentences extracted from the Internet are 2.54 times longer than Wikipedia ones. When Japanese Wikipedia is checked[130], it reveals that dots are also missing. I am not able to explain, why this happened.

4.4.3 Conditional Entropy and Conditional Perplexity

The conditional entropy is presented in Table C.3 and visualized in Figure 4.5. The average ratio between the conditional perplexity computed for the Wiki

[129] `http://ur.wikipedia.org/wiki/`
[130] `http://ja.wikipedia.org/`

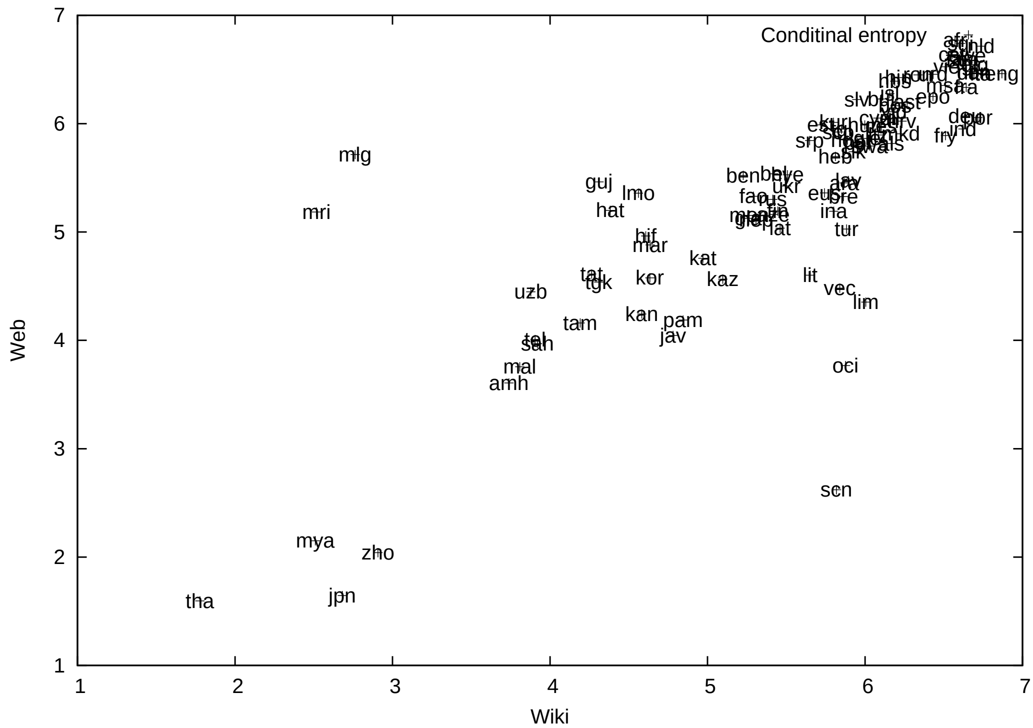


Figure 4.5: Wiki vs Web — conditional entropy

Corpus and the Web Corpus is 0.98, which signalizes, that the downloaded quite corresponds to the data retrieved from Wikipedia. On the one side are Maori (mri) and Malagasy (mlg) with ratio over 2 and on the other side Sicilian with ratio bellow 0.5.

The conditional perplexity is presented in Table C.4 and visualized in Figure 4.6.

4.4.4 Conclusions

All outliers have in common, that they are either from minor languages, such as Occitan (oci), Sicilian (scn), Maori (mri), Malagasy (mlg), for which low quality texts were collected, or they are written in non-latin scripts, such as Japanese (jpn), Chinese (zho), Nepali (nep), which are sensitive to preprocessing.

When different clustering algorithms were applied, then languages in same clusters does not have too much common properties.

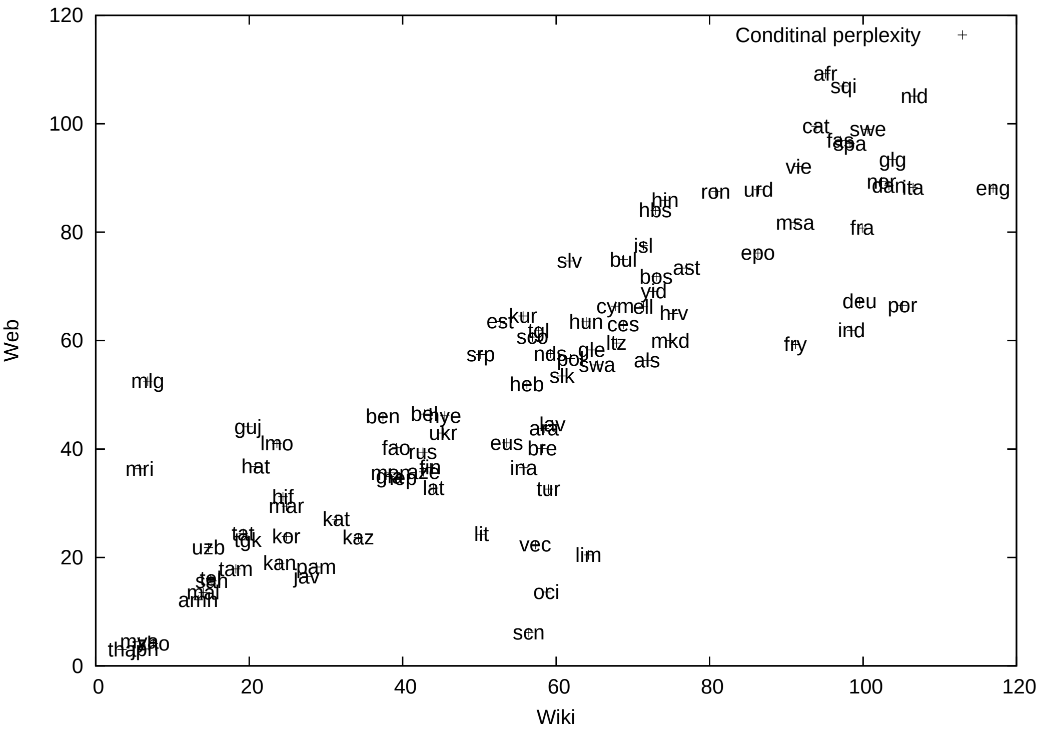


Figure 4.6: Wiki vs Web — conditional perplexity

4.5 Internet Size

The W2C Corpus was then used to estimate the Internet size in gigabytes of texts and number of pages. Table 4.7 presents size of the Internet in gigabytes and Table 4.8 in million of pages. The total size of all texts was estimated to 3.2PB and the total number of pages to 1.3 trillion pages. Pages written in English occupy 57% of the Internet, Russian 9%, Spanish 7%, French 4% and Arabic as well as German 2%.

Estimated numbers are much higher than they should be according to available information. This overestimation is caused by the search query selection, where phrases from Wikipedia were used, for example when 15 word English query is used hundreds of results are return.

ISO	Size	Part	ISO	Size	Part	ISO	Size	Part
afr	92.303	0.000	hif	0.040	0.000	nor	6215.762	0.001
als	0.036	0.000	hin	2651.988	0.000	oci	13.364	0.000
amh	0.004	0.000	hrv	12190.795	0.003	pam	0.004	0.000
ara	89416.080	0.027	hun	3.769	0.000	pol	51055.931	0.015
ast	9.258	0.000	hye	166.321	0.000	por	45789.010	0.014
aze	602.170	0.000	ina	0.038	0.000	ron	7717.890	0.002
bel	269.672	0.000	ind	51542.119	0.016	rus	313035.213	0.097
ben	0.024	0.000	isl	3331.202	0.001	sah	0.012	0.000
bos	2452.213	0.000	ita	27496.694	0.008	scn	0.265	0.000
bre	0.056	0.000	jav	0.016	0.000	sco	518.982	0.000
bul	15172.081	0.004	jpn	1690.396	0.000	slk	10.676	0.000
cat	9771.115	0.003	kan	7.579	0.000	slv	7.230	0.000
ces	8663.921	0.002	kat	2065.458	0.000	spa	208703.996	0.064
cym	92.672	0.000	kaz	43.548	0.000	sqi	1252.158	0.000
dan	6228.309	0.001	kor	0.368	0.000	srp	1347.320	0.000
deu	84322.440	0.026	kur	1.729	0.000	swa	0.096	0.000
ell	27149.268	0.008	lat	0.589	0.000	swe	12886.093	0.004
eng	1838631.170	0.571	lav	1159.025	0.000	tam	0.080	0.000
epo	46.632	0.000	lim	0.013	0.000	tat	9.852	0.000
est	630.203	0.000	lit	10.901	0.000	tel	0.028	0.000
eus	2.777	0.000	lmo	0.004	0.000	tgk	0.012	0.000
fao	0.999	0.000	ltz	1.878	0.000	tgl	174.601	0.000
fas	39487.743	0.012	mal	19.729	0.000	tha	0.116	0.000
fin	1952.586	0.000	mar	0.072	0.000	tur	25279.695	0.007
fra	150028.022	0.046	mkd	2373.767	0.000	ukr	21913.459	0.006
fry	0.425	0.000	mlg	0.000	0.000	urd	371.984	0.000
gla	0.065	0.000	mon	401.265	0.000	uzb	0.080	0.000
gle	25.420	0.000	mri	0.086	0.000	vec	0.075	0.000
glg	1174.611	0.000	msa	22167.686	0.006	vie	50267.527	0.015
guj	0.433	0.000	mya	0.084	0.000	yid	0.211	0.000
hat	0.000	0.000	nds	0.020	0.000	zho	12402.360	0.003
hbs	7632.790	0.002	nep	0.005	0.000			
heb	11154.272	0.003	nld	36260.232	0.011			

Table 4.7: Internet — size in GB

ISO	Pages	Part	ISO	Pages	Part	ISO	Pages	Part
afr	117.781	0.000	hif	0.003	0.000	nor	5333.773	0.003
als	0.006	0.000	hin	227.653	0.000	oci	22.174	0.000
amh	0.005	0.000	hrv	10729.230	0.007	pam	0.003	0.000
ara	26625.484	0.019	hun	1.938	0.000	pol	22574.016	0.016
ast	42.338	0.000	hye	63.672	0.000	por	32832.643	0.024
aze	184.932	0.000	ina	0.901	0.000	ron	3270.138	0.002
bel	80.743	0.000	ind	22146.039	0.016	rus	63948.004	0.047
ben	0.006	0.000	isl	813.613	0.000	sah	0.005	0.000
bos	1617.746	0.001	ita	16645.993	0.012	scn	2.021	0.000
bre	0.026	0.000	jav	0.015	0.000	sco	485.694	0.000
bul	3413.059	0.002	jpn	87.777	0.000	slk	4.916	0.000
cat	4016.530	0.002	kan	0.456	0.000	slv	7.719	0.000
ces	4416.463	0.003	kat	639.687	0.000	spa	54071.089	0.040
cym	595.061	0.000	kaz	41.593	0.000	sqi	240.534	0.000
dan	3706.546	0.002	kor	0.047	0.000	srp	319.285	0.000
deu	53575.887	0.039	kur	0.792	0.000	swa	0.042	0.000
ell	6163.909	0.004	lat	0.926	0.000	swe	5414.419	0.004
eng	779025.248	0.578	lav	736.307	0.000	tam	0.009	0.000
epo	60.814	0.000	lim	0.103	0.000	tat	6.454	0.000
est	301.061	0.000	lit	7.997	0.000	tel	0.012	0.000
eus	8.040	0.000	lmo	0.003	0.000	tgk	0.006	0.000
fao	0.606	0.000	ltz	1.231	0.000	tgl	318.172	0.000
fas	8980.481	0.006	mal	1.097	0.000	tha	0.009	0.000
fin	1356.983	0.001	mar	0.034	0.000	tur	26286.978	0.019
fra	85847.997	0.063	mkd	950.037	0.000	ukr	9853.784	0.007
fry	0.639	0.000	mlg	0.000	0.000	urd	69.210	0.000
gla	0.051	0.000	mon	70.244	0.000	uzb	0.041	0.000
gle	16.184	0.000	mri	0.114	0.000	vec	0.243	0.000
glg	578.176	0.000	msa	11391.889	0.008	vie	45520.137	0.033
guj	0.050	0.000	mya	0.003	0.000	yid	0.048	0.000
hat	0.009	0.000	nds	0.004	0.000	zho	1261.691	0.000
hbs	2697.391	0.002	nep	0.011	0.000			
heb	8187.467	0.006	nld	18858.003	0.014			

Table 4.8: Internet — page counts

5. Conclusions

The Web Corpus 'W2C' consists of at least 10 million words for each of the included 97 languages out of which 63 contain more than 100 million words. For the purpose of corpus constructing tools for collecting metadata, building corpus from Wikipedia, language recognition and distributed crawling, duplicity reduction and statistical analysis were developed.

The language metadata is automatically extracted from Ethnologue and Wikipedia and stored in the database. The collected metadata is used and extended by all the components.

Wikipedia was used as the source for the initial corpus. The Wiki Corpus was constructed from Wikipedias with at least 5 thousand articles. The Wiki Corpus contains 20 thousand articles (or as many as available) for 122 languages. This corpus served for training and testing of a language recognizer, as well as a baseline for comparison with the web corpus.

A language recognizer for 122 languages was developed. The recognizer was able to achieve 0.885 accuracy (median 0.982). The model used in the recognizer was specially tuned for corpus building, which may have decreased its overall accuracy.

The Web Corpus was build from more than 100 million pages, which were downloaded in the computer laboratory on 35 computers with a total execution time over a year. The computer laboratory is used by faculty students, so the developed solution had to be able to recover from failures, caused by memory exhaustion and computer restarts.

The raw corpus of downloaded data contained at least 10 million words for each of the 106 languages included at that time and for 96 of them more than 100 million words. Because the quality of the resulting corpus was important too, only 97 languages remained with size higher than 10 million of words after duplicity reduction. The total corpus size is 10.5 billion words, almost 90GB of texts.

Both corpora were statistically analysed and compared.

The unclear legal status of downloaded material does not allow publishing of the Web Corpus on the Internet, so the W2C-97-10 corpus with 10M word for 97 languages was released for internal usage only.

One of the goals, building corpus for hundreds of languages, was not achieved. There are only around 60 languages, that are used in industrialized countries, making them possible to be downloaded without any special effort. For the next 30 languages, it was possible to build the corpus with the required size, but a lot of duplicate content was downloaded. It would be possible to achieve the quota one hundred of languages for the cost of decreasing corpus quality. Downloading hundreds of languages would require collecting initial corpus for this amount of languages, which are not easily accessible. If this initial corpora would be available, highly specialized language recognizer for each language would be necessary, because only very short text fragments would be analysed. And even if this recognizer would be available, it still could not be possible to automatically download the texts, because they are not available on-line.

All downloaded data, more than 4.5TB, were preserved, so that they can be investigated further and more information about real language usage can be revealed, such as distribution of encodings or scripts for each language. Different tools for text extraction, language recognition and duplicity detection may be plugged-in. If the text extractor could extract texts segments instead of complete pages, it would be possible to increase corpus size for minor languages. A different set-up of existing tools allows constructing corpora for many purposes, from the hight quality ones for manual usage to the low quality ones for machine processing. Also, a specialized single topic corpus could be compiled.

Also, many partial topics can be investigated in a more detailed way. For example the language recognition problem, where dozens of parameters and methods combinations were ad-hoc tested, requires more rigorous approach. The text extraction problem could be studied as a complex problem together with duplicity reduction. Where a much simpler extractor does not remove all boilerplate code, but with duplicity reduction on line level, this boilerplate code is removed. All these methods could also be investigated from a performance view, where simpler methods could save weeks of computation for the cost of slightly decreased quality.

The W2C Corpus is a unique data source for linguists, because it outclasses all published works both in the size of collected material and the number of covered languages. The collected data may be used for comparative analysis of related languages, building language models for various applications such as machine translation, speech recognition, spell checking, etc.

A. DVD Content

All source codes, the W2C Corpus and text of this thesis are available on the DVD. The DVD has the following directory structure:

- `corpus` — the W2C-97-10 Corpus — 10 million words for 97 languages
- `text` — text of this thesis in version for viewing and printing
- `source` — contains the tarball with source code

The source codes extracted from the tarball has the following content:

- `checkRequirements.sh` — checks, whether all required programs and libraries are available
- `bin` — symlinks for scripts located in `pipes`, `visualizations` and `tools`
- `builder` — directory containing the W2C Builder
- `data` — retrieved and generated files, all scripts are storing results to this folder
- `experiments` — scripts for experiments
- `pipes` — scripts that read standard input and modified output print out to standard output
- `scripts` — scripts that are useful on specific environment — the computer laboratory or the server ufallab
- `tools` — scripts for specific tasks
  - `aspellCoverage` — scripts for computing coverage of an aspell dictionary
  - `crawlerSimple` — simple crawler for downloading web pages
  - `ethnologueParser` — scripts for parsing the Ethnologue website
  - `fillLangDB` — scripts for filling the metadata database
  - `internetSize` — script for estimating the Internet size
  - `langDetect` — scripts for training and testing language recognizer
  - `langList` — scripts for parsing Wikipedia
  - `search` — scripts for retrieving results from search engines
  - `utils` — scripts with miscellaneous purposes
  - `webAPI` — command line client to the database
  - `wikiCorpora` — scripts for building the corpus from the Wikipedia dumps

  - `wikiExternalLinks` — scripts for extraction external links from Wikipedia
  - `wikiMiniCorpora` — scripts for building the corpus from the Wikipedia pages

- `visualizations` — scripts that are used to visualization, formatting or analysis of the input file

B. List of Languages

All information are automatically extracted from ethnologue[131].

Column — *Lang*: ISO 639-3 code, *Name*: language name, *Pop*: population in thousands, *WO*: Word Order typology and *Script*: used script

Table B.1: List of Languages

ISO	Name	Pop	Type	Classification
afr	Afrikaans	4934	Liv	Indo-European, Germanic, West
als	Tosk Albanian	3035	Liv	Indo-European, Albanian, Tosk
amh	Amharic	17528	Liv	Afro-Asiatic, Semitic, South
ara	Arabic	221002	Liv	Afro-Asiatic, Semitic, Central
arg	Aragonese	2000	Liv	Indo-European, Italic, Romance
arz	Egyptian Arabic	53990	Liv	Afro-Asiatic, Semitic, Central
ast	Asturian	125	Liv	Indo-European, Italic, Romance
aze	Azerbaijani	19147	Liv	Altaic, Turkic, Southern
bcl	Central Bicolano	2500	Liv	Austronesian, Malayo-Polynesian, Philipp
bel	Belarusian	8618	Liv	Indo-European, Slavic, East
ben	Bengali	181272	Liv	Indo-European, Indo-Iranian, Indo-Aryan
bos	Bosnian	2203	Liv	Indo-European, Slavic, South
bpy	Bishnupriya	115	Liv	Indo-European, Indo-Iranian, Indo-Aryan
bre	Breton	500	Liv	Indo-European, Celtic, Insular
bug	Buginese	3500	Liv	Austronesian, Malayo-Polynesian, South S
bul	Bulgarian	9097	Liv	Indo-European, Slavic, South
cat	Catalan	11530	Liv	Indo-European, Italic, Romance
ceb	Cebuano	15807	Liv	Austronesian, Malayo-Polynesian, Philipp
ces	Czech	9490	Liv	Indo-European, Slavic, West
chv	Chuvash	1674	Liv	Altaic, Turkic, Bolgar
cos	Corsican	402	Liv	Indo-European, Italic, Romance
cym	Welsh	537	Liv	Indo-European, Celtic, Insular
dan	Danish	5581	Liv	Indo-European, Germanic, North
deu	German	90294	Liv	Indo-European, Germanic, West
diq	Dimli	1000	Liv	Indo-European, Indo-Iranian, Iranian
ell	Modern Greek	13084	Liv	Indo-European, Greek, Attic
eng	English	328008	Liv	Indo-European, Germanic, West
epo	Esperanto	0	Con	Constructed language
est	Estonian	1048	Liv	Uralic, Finnic
eus	Basque	658	Liv	Basque
fao	Faroese	48	Liv	Indo-European, Germanic, North
fas	Persian	31381	Liv	Indo-European, Indo-Iranian, Iranian

Continued on Next Page...

[131] http://ethnologue.org

ISO	Name	Pop	Type	Classification
fin	Finnish	5009	Liv	Uralic, Finnic
fra	French	67838	Liv	Indo-European, Italic, Romance
fry	Western Frisian	467	Liv	Indo-European, Germanic, West
gan	Gan Chinese	20600	Liv	Sino-Tibetan, Chinese
gla	Scottish Gaelic	66	Liv	Indo-European, Celtic, Insular
gle	Irish	391	Liv	Indo-European, Celtic, Insular
glg	Galician	3185	Liv	Indo-European, Italic, Romance
glk	Gilaki	3270	Liv	Indo-European, Indo-Iranian, Iranian
guj	Gujarati	46493	Liv	Indo-European, Indo-Iranian, Indo-Aryan
hat	Haitian	7701	Liv	Creole, French based
hbs	Serbo-Croatian	16351	Liv	Indo-European, Slavic, South
heb	Hebrew	5316	Liv	Afro-Asiatic, Semitic, Central
hif	Fiji Hindi	380	Liv	Indo-European, Indo-Iranian, Indo-Aryan
hin	Hindi	181676	Liv	Indo-European, Indo-Iranian, Indo-Aryan
hrv	Croatian	5546	Liv	Indo-European, Slavic, South
hsb	Upper Sorbian	18	Liv	Indo-European, Slavic, West
hun	Hungarian	12501	Liv	Uralic
hye	Armenian	6376	Liv	Indo-European, Armenian
ido	Ido	0	Con	
ina	Interlingua	0	Con	
ind	Indonesian	23187	Liv	Austronesian, Malayo-Polynesian, Malayo-
isl	Icelandic	238	Liv	Indo-European, Germanic, North
ita	Italian	61696	Liv	Indo-European, Italic, Romance
jav	Javanese	84608	Liv	Austronesian, Malayo-Polynesian, Javanes
jpn	Japanese	122080	Liv	Japonic
kan	Kannada	35327	Liv	Dravidian, Southern, Tamil-Kannada
kat	Georgian	4255	Liv	Kartvelian, Georgian
kaz	Kazakh	8331	Liv	Altaic, Turkic, Western
kor	Korean	66305	Liv	Language isolate
kur	Kurdish	16025	Liv	Indo-European, Indo-Iranian, Iranian
lat	Latin	0	Anc	Indo-European, Italic, Latino-Faliscan
lav	Latvian	1504	Liv	Indo-European, Baltic, Eastern
lim	Limburgan	1300	Liv	Indo-European, Germanic, West
lit	Lithuanian	3154	Liv	Indo-European, Baltic, Eastern
lmo	Lombard	9133	Liv	Indo-European, Italic, Romance
ltz	Luxembourgish	320	Liv	Indo-European, Germanic, West
mal	Malayalam	35893	Liv	Dravidian, Southern, Tamil-Kannada
mar	Marathi	68061	Liv	Indo-European, Indo-Iranian, Indo-Aryan
mkd	Macedonian	2113	Liv	Indo-European, Slavic, South
mlg	Malagasy	14736	Liv	Austronesian, Malayo-Polynesian, Greater
mon	Mongolian	5720	Liv	Altaic, Mongolic, Eastern
mri	Maori	60	Liv	Austronesian, Malayo-Polynesian, Central

Continued on Next Page…

ISO	Name	Pop	Type	Classification
msa	Malay	39144	Liv	Austronesian, Malayo-Polynesian, Malayo-
mya	Burmese	32319	Liv	Sino-Tibetan, Tibeto-Burman, Lolo-Burmes
nap	Neapolitan	7050	Liv	Indo-European, Italic, Romance
nds	Low German	1	Liv	Indo-European, Germanic, West
nep	Nepali	13875	Liv	Indo-European, Indo-Iranian, Indo-Aryan
new	Newari	839	Liv	Sino-Tibetan, Tibeto-Burman, Himalayish
nld	Dutch	21730	Liv	Indo-European, Germanic, West
nno	Norwegian Nynorsk	0	Liv	
nor	Norwegian	4640	Liv	Indo-European, Germanic, North
oci	Occitan	2048	Liv	Indo-European, Italic, Romance
oss	Ossetian	641	Liv	Indo-European, Indo-Iranian, Iranian
pam	Pampanga	1905	Liv	Austronesian, Malayo-Polynesian, Philipp
pms	Piemontese	3110	Liv	Indo-European, Italic, Romance
pnb	Western Panjabi	62648	Liv	Indo-European, Indo-Iranian, Indo-Aryan
pol	Polish	39990	Liv	Indo-European, Slavic, West
por	Portuguese	177981	Liv	Indo-European, Italic, Romance
que	Quechua	10098	Liv	Quechuan, Quechua II, C
ron	Romanian	23351	Liv	Indo-European, Italic, Romance
rus	Russian	143553	Liv	Indo-European, Slavic, East
sah	Yakut	443	Liv	Altaic, Turkic, Northern
scn	Sicilian	4830	Liv	Indo-European, Italic, Romance
sco	Scots	200	Liv	Indo-European, Germanic, West
sgs	Samogitian	0	Liv	
slk	Slovak	5019	Liv	Indo-European, Slavic, West
slv	Slovenian	1909	Liv	Indo-European, Slavic, South
spa	Spanish	328518	Liv	Indo-European, Italic, Romance
sqi	Albanian	5825	Liv	Indo-European, Albanian, Gheg
srp	Serbian	7020	Liv	Indo-European, Slavic, South
sun	Sundanese	34000	Liv	Austronesian, Malayo-Polynesian, Malayo-
swa	Swahili	730	Liv	Niger-Congo, Atlantic-Congo, Volta-Congo
swe	Swedish	8311	Liv	Indo-European, Germanic, North
tam	Tamil	65675	Liv	Dravidian, Southern, Tamil-Kannada
tat	Tatar	6496	Liv	Altaic, Turkic, Western
tel	Telugu	69758	Liv	Dravidian, South-Central, Telugu
tgk	Tajik	4457	Liv	Indo-European, Indo-Iranian, Iranian
tgl	Tagalog	23853	Liv	Austronesian, Malayo-Polynesian, Philipp
tha	Thai	20362	Liv	Tai-Kadai, Kam-Tai, Be-Tai
tur	Turkish	50750	Liv	Altaic, Turkic, Southern
ukr	Ukrainian	37029	Liv	Indo-European, Slavic, East
urd	Urdu	60586	Liv	Indo-European, Indo-Iranian, Indo-Aryan
uzb	Uzbek	20250	Liv	Altaic, Turkic, Eastern
vec	Venetian	6230	Liv	Indo-European, Italic, Romance

Continued on Next Page…

ISO	Name	Pop	Type	Classification
vie	Vietnamese	68634	Liv	Austro-Asiatic, Mon-Khmer, Viet-Muong
vol	Volapük	0	Con	
war	Waray	2570	Liv	Austronesian, Malayo-Polynesian, Philipp
wln	Walloon	1120	Liv	Indo-European, Italic, Romance
yid	Yiddish	2255	Liv	Indo-European, Germanic, West
yor	Yoruba	19380	Liv	Niger-Congo, Atlantic-Congo, Volta-Congo
zho	Chinese	1212515	Liv	Sino-Tibetan, Chinese

C. Wiki vs Web

This appendix contains raw data for comparing the Wiki Corpus and the W2C Corpus.

- Average Word Length (C.1)
- Average Sentence Length (C.2
- Conditional Entropy (C.3)
- Conditional Perplexity (C.4)

ISO	Wiki	Web	R	ISO	Wiki	Web	R	ISO	Wiki	Web	R
afr	9.30	9.44	1.01	hif	6.41	5.99	0.93	nor	9.88	9.97	1.01
als	9.23	9.69	1.05	hin	6.95	6.67	0.96	oci	7.64	10.31	1.35
amh	4.78	5.19	1.09	hrv	8.33	8.65	1.04	pam	7.39	8.43	1.14
ara	6.33	6.94	1.10	hun	9.67	10.29	1.06	pol	8.88	9.55	1.08
ast	7.88	8.50	1.08	hye	8.92	9.31	1.04	por	7.99	8.43	1.06
aze	8.53	9.24	1.08	ina	7.76	8.56	1.10	ron	8.10	8.83	1.09
bel	8.56	8.80	1.03	ind	7.48	8.42	1.13	rus	8.97	9.68	1.08
ben	7.86	7.60	0.97	isl	9.40	9.27	0.99	sah	8.45	9.27	1.10
bos	8.31	8.50	1.02	ita	8.16	8.96	1.10	scn	7.99	10.40	1.30
bre	7.21	8.70	1.21	jav	7.39	9.66	1.31	sco	7.10	7.94	1.12
bul	8.35	8.44	1.01	jpn	7.40	12.30	1.66	slk	8.52	8.81	1.03
cat	7.79	8.61	1.11	kan	10.20	9.59	0.94	slv	8.12	8.66	1.07
ces	8.35	8.72	1.04	kat	8.84	9.12	1.03	spa	8.17	9.04	1.11
cym	7.48	8.41	1.12	kaz	8.62	9.10	1.05	sqi	7.91	8.07	1.02
dan	9.82	10.19	1.04	kor	4.25	5.07	1.19	srp	8.10	8.29	1.02
deu	10.89	10.78	0.99	kur	7.14	7.46	1.04	swa	8.13	8.54	1.05
ell	8.58	9.63	1.12	lat	8.54	9.88	1.16	swe	9.85	9.99	1.01
eng	7.55	8.44	1.12	lav	8.51	9.12	1.07	tam	10.65	10.62	1.00
epo	8.26	10.18	1.23	lim	8.66	9.22	1.06	tat	8.01	8.34	1.04
est	9.85	10.27	1.04	lit	8.79	9.51	1.08	tel	9.01	9.15	1.02
eus	9.06	10.57	1.17	lmo	7.07	7.99	1.13	tgk	7.33	7.84	1.07
fao	8.57	8.86	1.03	ltz	9.45	9.72	1.03	tgl	7.80	8.44	1.08
fas	6.65	6.96	1.05	mal	12.08	12.37	1.02	tha	28.14	23.65	0.84
fin	11.19	11.04	0.99	mar	8.04	7.90	0.98	tur	8.86	9.46	1.07
fra	7.84	8.71	1.11	mkd	8.30	8.61	1.04	ukr	8.74	9.50	1.09
fry	9.12	10.38	1.14	mlg	7.18	8.46	1.18	urd	5.92	6.43	1.09
gla	7.45	7.69	1.03	mon	7.75	7.91	1.02	uzb	8.35	8.88	1.06
gle	8.16	8.55	1.05	mri	6.97	7.18	1.03	vec	7.45	7.84	1.05
glg	8.04	8.92	1.11	msa	7.42	10.09	1.36	vie	6.08	6.03	0.99
guj	7.40	7.28	0.98	mya	14.28	13.28	0.93	yid	7.42	7.27	0.98
hat	6.47	6.76	1.05	nds	9.32	10.13	1.09	zho	6.77	7.76	1.15
hbs	8.29	8.44	1.02	nep	7.66	8.05	1.05				
heb	6.31	6.60	1.05	nld	9.59	9.98	1.04				

Table C.1: Wiki vs Web — average word length

ISO	Wiki	Web	R	ISO	Wiki	Web	R	ISO	Wiki	Web	R
afr	19.86	17.68	0.89	hif	15.34	18.87	1.23	nor	15.71	13.87	0.88
als	15.45	15.06	0.97	hin	63.36	34.27	0.54	oci	21.14	22.80	1.08
amh	65.34	57.07	0.87	hrv	14.70	17.05	1.16	pam	17.13	18.01	1.05
ara	24.45	30.85	1.26	hun	13.93	15.23	1.09	pol	14.92	14.02	0.94
ast	21.91	22.24	1.01	hye	64.22	53.80	0.84	por	22.32	16.81	0.75
aze	13.07	13.57	1.04	ina	19.59	17.55	0.90	ron	20.06	19.68	0.98
bel	12.10	13.83	1.14	ind	17.37	14.75	0.85	rus	15.50	15.83	1.02
ben	175.91	70.08	0.40	isl	14.89	16.44	1.10	sah	9.39	11.33	1.21
bos	14.97	17.51	1.17	ita	25.55	18.59	0.73	scn	21.23	24.46	1.15
bre	20.12	20.57	1.02	jav	14.73	14.52	0.99	sco	19.34	16.45	0.85
bul	14.95	16.04	1.07	jpn	24.62	59.51	2.42	slk	13.92	14.82	1.06
cat	25.11	22.76	0.91	kan	13.42	9.80	0.73	slv	15.72	15.77	1.00
ces	14.88	14.73	0.99	kat	11.69	13.82	1.18	spa	24.92	22.81	0.92
cym	18.52	20.53	1.11	kaz	10.89	12.40	1.14	sqi	20.75	22.52	1.09
dan	16.16	16.70	1.03	kor	14.01	13.58	0.97	srp	14.41	17.89	1.24
deu	16.62	15.47	0.93	kur	14.51	18.44	1.27	swa	17.16	17.70	1.03
ell	19.60	18.68	0.95	lat	14.79	18.82	1.27	swe	16.97	15.01	0.88
eng	21.56	19.11	0.89	lav	12.37	15.70	1.27	tam	10.74	10.73	1.00
epo	16.37	17.92	1.09	lim	16.53	15.48	0.94	tat	10.79	11.89	1.10
est	11.75	13.07	1.11	lit	10.58	13.60	1.28	tel	11.25	9.48	0.84
eus	13.00	14.22	1.09	lmo	18.17	18.18	1.00	tgk	13.46	19.30	1.43
fao	13.43	15.27	1.14	ltz	15.62	17.26	1.11	tgl	18.73	14.64	0.78
fas	21.19	25.18	1.19	mal	9.28	9.06	0.98	tha	23.93	18.28	0.76
fin	12.12	12.53	1.03	mar	11.56	11.62	1.01	tur	13.49	14.31	1.06
fra	23.83	22.95	0.96	mkd	18.06	16.78	0.93	ukr	13.32	13.67	1.03
fry	15.86	15.03	0.95	mlg	17.82	20.71	1.16	urd	429.52	151.94	0.35
gla	18.86	20.55	1.09	mon	16.12	14.58	0.90	uzb	13.44	14.43	1.07
gle	20.62	19.77	0.96	mri	20.84	20.55	0.99	vec	22.64	16.71	0.74
glg	21.65	19.55	0.90	msa	17.43	15.95	0.92	vie	27.05	25.25	0.93
guj	17.32	15.99	0.92	mya	147.34	17.86	0.12	yid	23.35	24.04	1.03
hat	12.39	17.82	1.44	nds	14.10	14.76	1.05	zho	18.10	12.62	0.70
hbs	15.49	16.68	1.08	nep	62.44	107.89	1.73				
heb	17.12	16.15	0.94	nld	17.66	15.36	0.87				

Table C.2: Wiki vs Web — average sentence length

ISO	Wiki	Web	R	ISO	Wiki	Web	R	ISO	Wiki	Web	R
afr	6.57	6.77	1.03	hif	4.61	4.96	1.08	nor	6.68	6.48	0.97
als	6.17	5.82	0.94	hin	6.21	6.42	1.03	oci	5.88	3.77	0.64
amh	3.74	3.61	0.96	hrv	6.24	6.02	0.97	pam	4.85	4.19	0.86
ara	5.87	5.45	0.93	hun	6.00	5.99	1.00	pol	5.95	5.82	0.98
ast	6.27	6.20	0.99	hye	5.51	5.53	1.00	por	6.72	6.06	0.90
aze	5.42	5.16	0.95	ina	5.80	5.19	0.89	ron	6.34	6.45	1.02
bel	5.42	5.54	1.02	ind	6.62	5.95	0.90	rus	5.41	5.30	0.98
ben	5.23	5.52	1.06	isl	6.16	6.28	1.02	sah	3.92	3.97	1.01
bos	6.19	6.17	1.00	ita	6.74	6.46	0.96	scn	5.82	2.62	0.45
bre	5.86	5.33	0.91	jav	4.78	4.05	0.85	sco	5.83	5.92	1.02
bul	6.10	6.23	1.02	jpn	2.68	1.64	0.61	slk	5.93	5.74	0.97
cat	6.55	6.64	1.01	kan	4.58	4.24	0.93	slv	5.95	6.22	1.05
ces	6.10	5.98	0.98	kat	4.97	4.76	0.96	spa	6.62	6.59	1.00
cym	6.08	6.05	1.00	kaz	5.10	4.57	0.90	sqi	6.61	6.74	1.02
dan	6.69	6.47	0.97	kor	4.63	4.58	0.99	srp	5.65	5.84	1.03
deu	6.64	6.07	0.91	kur	5.80	6.01	1.04	swa	6.03	5.79	0.96
ell	6.16	6.05	0.98	lat	5.46	5.04	0.92	swe	6.65	6.63	1.00
eng	6.87	6.46	0.94	lav	5.90	5.48	0.93	tam	4.19	4.16	0.99
epo	6.43	6.25	0.97	lim	6.01	4.35	0.73	tat	4.26	4.61	1.08
est	5.72	5.99	1.05	lit	5.65	4.60	0.81	tel	3.91	4.01	1.03
eus	5.74	5.36	0.93	lmo	4.56	5.36	1.17	tgk	4.31	4.54	1.05
fao	5.29	5.33	1.01	ltz	6.08	5.90	0.97	tgl	5.85	5.95	1.02
fas	6.60	6.60	1.00	mal	3.81	3.76	0.99	tha	1.78	1.60	0.90
fin	5.45	5.19	0.95	mar	4.64	4.88	1.05	tur	5.88	5.03	0.85
fra	6.64	6.34	0.95	mkd	6.23	5.91	0.95	ukr	5.50	5.42	0.99
fry	6.51	5.89	0.90	mlg	2.76	5.71	2.07	urd	6.43	6.46	1.00
gla	5.26	5.13	0.97	mon	5.27	5.15	0.98	uzb	3.88	4.45	1.15
gle	6.01	5.87	0.98	mri	2.52	5.18	2.06	vec	5.84	4.48	0.77
glg	6.70	6.54	0.98	msa	6.51	6.35	0.98	vie	6.52	6.53	1.00
guj	4.31	5.46	1.27	mya	2.51	2.15	0.86	yid	6.18	6.11	0.99
hat	4.38	5.20	1.19	nds	5.89	5.85	0.99	zho	2.91	2.04	0.70
hbs	6.19	6.39	1.03	nep	5.31	5.11	0.96				
heb	5.81	5.70	0.98	nld	6.74	6.72	1.00				

Table C.3: Wiki vs Web — conditional entropy

ISO	Wiki	Web	R	ISO	Wiki	Web	R	ISO	Wiki	Web	R
afr	95.11	109.23	1.15	hif	24.39	31.20	1.28	nor	102.41	89.33	0.87
als	71.85	56.38	0.78	hin	74.21	85.88	1.16	oci	58.75	13.62	0.23
amh	13.36	12.19	0.91	hrv	75.35	65.02	0.86	pam	28.75	18.23	0.63
ara	58.43	43.82	0.75	hun	63.92	63.45	0.99	pol	61.89	56.68	0.92
ast	77.01	73.38	0.95	hye	45.50	46.13	1.01	por	105.15	66.52	0.63
aze	42.73	35.79	0.84	ina	55.77	36.54	0.66	ron	80.81	87.48	1.08
bel	42.85	46.44	1.08	ind	98.50	61.91	0.63	rus	42.65	39.43	0.92
ben	37.44	45.99	1.23	isl	71.38	77.52	1.09	sah	15.13	15.67	1.04
bos	73.04	71.78	0.98	ita	106.53	88.23	0.83	scn	56.44	6.15	0.11
bre	58.20	40.13	0.69	jav	27.51	16.52	0.60	sco	56.92	60.66	1.07
bul	68.76	74.93	1.09	jpn	6.42	3.13	0.49	slk	60.78	53.48	0.88
cat	93.88	99.56	1.06	kan	23.97	18.94	0.79	slv	61.75	74.68	1.21
ces	68.75	62.96	0.92	kat	31.36	27.05	0.86	spa	98.30	96.35	0.98
cym	67.71	66.36	0.98	kaz	34.23	23.67	0.69	sqi	97.49	106.99	1.10
dan	103.38	88.58	0.86	kor	24.84	23.87	0.96	srp	50.18	57.47	1.15
deu	99.57	67.22	0.68	kur	55.71	64.56	1.16	swa	65.35	55.51	0.85
ell	71.38	66.23	0.93	lat	44.04	32.80	0.74	swe	100.65	98.98	0.98
eng	116.94	88.11	0.75	lav	59.52	44.49	0.75	tam	18.27	17.89	0.98
epo	86.30	76.16	0.88	lim	64.23	20.45	0.32	tat	19.22	24.45	1.27
est	52.70	63.52	1.21	lit	50.29	24.27	0.48	tel	14.98	16.08	1.07
eus	53.56	41.12	0.77	lmo	23.61	41.03	1.74	tgk	19.83	23.24	1.17
fao	39.18	40.23	1.03	ltz	67.87	59.57	0.88	tgl	57.74	61.81	1.07
fas	97.03	96.92	1.00	mal	14.00	13.51	0.96	tha	3.43	3.02	0.88
fin	43.60	36.57	0.84	mar	24.86	29.47	1.19	tur	59.01	32.61	0.55
fra	99.89	80.78	0.81	mkd	74.90	59.97	0.80	ukr	45.30	42.96	0.95
fry	91.19	59.31	0.65	mlg	6.79	52.52	7.74	urd	86.39	87.81	1.02
gla	38.30	34.93	0.91	mon	38.45	35.60	0.93	uzb	14.71	21.82	1.48
gle	64.65	58.29	0.90	mri	5.73	36.37	6.35	vec	57.29	22.35	0.39
glg	103.86	93.37	0.90	msa	91.16	81.73	0.90	vie	91.63	92.10	1.01
guj	19.86	44.10	2.22	mya	5.70	4.44	0.78	yid	72.74	69.10	0.95
hat	20.85	36.76	1.76	nds	59.24	57.60	0.97	zho	7.50	4.11	0.55
hbs	72.94	84.03	1.15	nep	39.63	34.64	0.87				
heb	56.19	51.90	0.92	nld	106.70	105.10	0.99				

Table C.4: Wiki vs Web — conditional perplexity

Bibliography

[BB98] Krishna Bharat and Andrei Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 379–388, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[BB01] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 26–33, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[BBFZ09] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43:209–226, 2009. 10.1007/s10579-009-9081-4.

[BFJ^{+}06] Andrei Broder, Marcus Fontura, Vanja Josifovski, Ravi Kumar, Rajeev Motwani, Shubha Nabar, Rina Panigrahy, Andrew Tomkins, and Ying Xu. Estimating corpus size via queries. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 594–603, New York, NY, USA, 2006. ACM.

[BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29:1157–1166, September 1997.

[BK06] Marco Baroni and Adam Kilgarriff. Large linguistically-processed web corpora for multiple languages. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, EACL '06, pages 87–90, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[BRSB00] A. Bharati, K. P. Rao, R. Sangal, and S. M. Bendre. Basic statistical analysis of corpus and cross comparison among corpora. *Technical Report of Indian Institute of Information Technology*, 2000.

[BYG07] Ziv Bar-Yossef and Maxim Gurevich. Efficient search engine measurements. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 401–410, New York, NY, USA, 2007. ACM.

[CT94] William B. Cavnar and John M. Trenkle. N-grambased text categorization. In *In Proc. of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[GS05] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 902–903, New York, NY, USA, 2005. ACM.

[Hay04] Katia Hayati. Language identification on the world wide web, 2004.

[HNP09] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[KRPA10] Adam Kilgarriff, Siva Reddy, Jan Pomikálek, and P. V. S. Avinesh. A corpus factory for many languages. In *Language Resources and Evaluation*, 2010.

[LL10] Jianguo Lu and Dingding Li. Estimating deep web data source size by capture—recapture method. *Inf. Retr.*, 13:70–95, February 2010.

[LM01] Shanjian Li and Katsuhiko Momoi. A composite approach to language/encoding detection. In *19th International Unicode Conference*, International Unicode Conference '01, 2001.

[MS05] Bruno Martins and Mário J. Silva. Language identification in web pages. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 764–768, New York, NY, USA, 2005. ACM.

[RG00] Paul Rayson and Roger Garside. Comparing corpora using frequency profiling. In *Proceedings of the workshop on Comparing corpora - Volume 9*, WCC '00, pages 1–6, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[Sca07] Kevin P. Scannell. *The Crúbadán Project: Corpus building for under-resourced languages*, volume 4 of *Cahiers du Cental*, pages 5–15. Louvain-la-Neuve, Belgium, 2007.

[Sha06] Serge Sharoff. Creating general-purpose corpora using automated search engine queries. In *WaCky! Working papers on the Web as Corpus. Gedit*, 2006.

[SR96] Penelope Sibun and Jeffrey C. Reynar. Language identification: Examining the issues, 1996.

[Wyn05] Martin Wynne. *Developing Linguistic Corpora: a Guide to Good Practice*, chapter Archiving, Distribution and Preservation, pages 71–78. Oxford: Oxbow Books, 2005. Available online, Accessed 2011-01-01.

[ZZY+08] Guo-Qing Zhang, Guo-Qiang Zhang, Qing-Feng Yang, Su-Qi Cheng, and Tao Zhou. Evolution of the internet and its cores. *New Journal of Physics*, 10(12):123027, 2008.

List of Tables

List of Figures