HARVESTING NATIONAL LANGUAGE TEXT CORPORA FROM THE WEB

Jānis Džeriņš, Kristaps Džonsons
Institute of Mathematics and Computer Science,
University of Latvia

Annotation

There is huge amount of textual data on the web. In this paper we show how a general purpose NLP tool can be used to grade linguistic quality of the texts gathered from the web. The described approach is of interest for "small languages", such as Latvian, with very limited NLP tools available. Massively parallel grid computing has been used to parse a rather complete Latvian web archive.

Keywords: text corpus, web crawling, NLP application, chunker

1. Introduction

A large quality text corpus is essential for linguistic research. Hand-selected literary texts from various domains are predominantly used. The downside of this approach is that such texts typically represent the literary language. An additional difficulty for smaller languages, such as Latvian, is that balanced, high-quality corpora are either not available or of very limited size.

There is no doubt that the Web can be used as a corpus (Kilgarriff and Grefenstette, 2003). Hand made text corpora usually contain a metadata that is not always available for Web texts. But in many cases that metadata is not even necessary.

General problems and techniques associated with corpora collection from the Web are discussed in (Bernardini et.al. 2006). One of the most involved problems is that of determining the quality of the text and its suitability for collection purposes. The innovation of our approach is the use of a [general purpose] NLP tools to select and grade the retrieved texts according to their linguistic qualities with respect to the target language.

In our case the target language is Latvian and the NLP tool used for deep inspection and grading of the retrieved texts was the first large-coverage syntactic parser of Latvian: the SemTi-Kamols project's parser (Bārzdiņs et al. 2007). This application recognizes syntactical, morphological and semantic constructs, thus allowing us to rate texts according to their relative processability.

2. Data collection

Dealing with multiple encodings is the first challenge. The following encodings are commonly used for the Latvian language:

(1) ISO8859-4 – original character set for Baltic languages, superseded by ISO8859-10, and later, by ISO8859-13.
(2) ISO8859-13 (also known as BalticRim) – character set used today. Encodings of Latvian accented characters are not compatible with ISO8859-4.

(3) Windows-1257 – a character set compatible with ISO8859-13, used in Windows™ operating system.

In the early days of World Wide Web, the web page authoring tools did not support these character sets (ISO8859-13 was created only in 1998). For instance, Netscape Composer™ only supported ISO8859-1; characters from other sets were converted to HTML *character entities*. When viewing such pages, it was important to have correct font set in the browser program – it converted *character entities* into their corresponding codes from the ISO8859-1 character set but since the font was for a different character set, different characters ended up being used. And the end result was that it all seemed to work, although only by accident.

Other problem with character sets is that HTML specification specifies[1] that content character set specified in the HTTP headers takes precedence over what is specified in the HTML document itself. And the recommended default is ISO8859-1. But in reality it does not make sense, because the server administrator cannot know in advance what character sets are used in users' files. And even if they did it would be an enormous effort for all Web server administrators to track HTML document sources and update configuration files with correct encodings.

What we end up with is that if we come to a document with a ISO8859-1 character set (specified in either HTTP headers or the document itself), that information is practically useless. Thus we have to employ heuristics in order to guess the correct character set.

We use a list of words that are very common in Latvian language texts and which do not contain any accented characters to determine whether the text in question really is in Latvian. At least 3% of words usually are from this list, which was selected from a corpus of texts known to be in Latvian:

ar bet bez desmit divi gan ir ja jo ka kas kaut nav ne par pieci tas tiek tur un uz vai viens visas visi

Here are the heuristics used by our crawler:

(1) We disobey the recommendation of W3C and treat the character set specification in the document as being more specific than the one specified by HTTP.

(2) If the character set is not specified or is specified to be ISO8859-1, and the text appears to be Latvian using the method just mentioned, we assume the ISO8859-13 encoding, since ISO8859-4 is obsolete. It would also be possible to try both of them and then use another set of frequently used words with accented characters to determine the correct one.

To further narrow down the set of crawled Web pages we only crawl pages from servers with IP addresses from Latvia and some manually configured Web sites that have been submitted by search engine users and verified to be in Latvian.

Our web crawler is one of the most popular in Latvia[2]. Even one of the biggest Latvian news portals[3] uses it as a search engine. What we have is around 4GB of text data, consisting of around 700k HTML-free texts from a few years back. Today the search engine contains around 2 million documents.

---

[1] HTML 4.01 specification, section 5.2.2, http://www.w3.org/TR/REC-html40/

[2] http://search.latnet.lv/

[3] http://www.tvnet.lv/search/

3. NLP application

The NLP tool we used is the first large-coverage Latvian text parser, developed in SemTi-Kamols project. This application can recognize a number of words and syntactic constructs but it is not complete. Nevertheless, this is presently the only wide-coverage

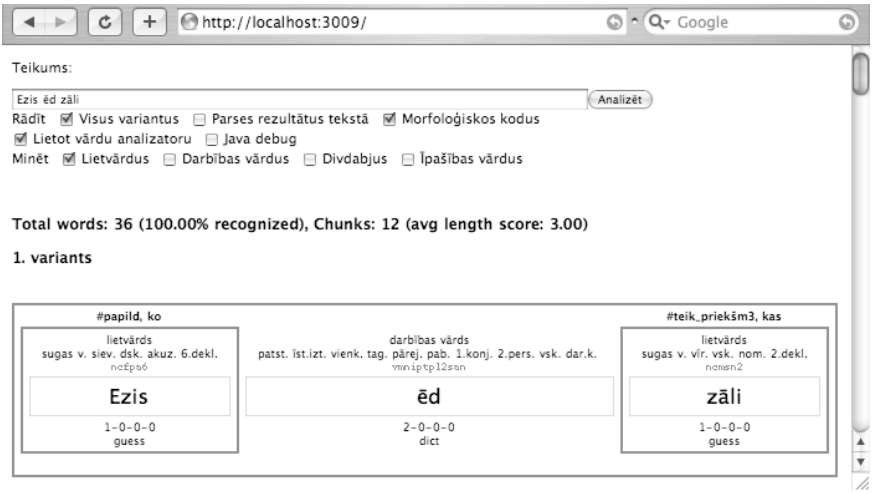

Figure 1. SemTi-Kamols user interface

tool available for deep analysis of Latvian. In fact the other purpose of this experiment was to see in what cases this tool is failing and what are the places that should be improved first.

Since we know the application is incomplete we run it in a mode we call

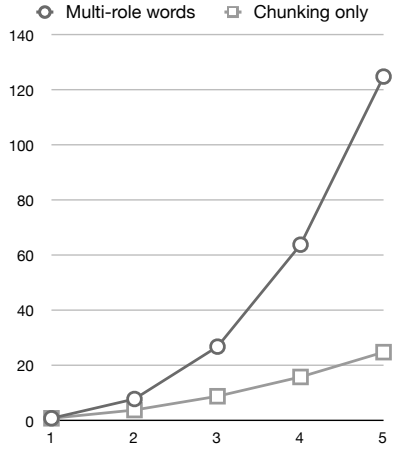

Figure 2. Chunker performance characteristics

chunking: For each sentence chunker tries all subsentences, and the longest such subsentence that has a valid parse is selected. This way there is lots of wasted

computation, but we get to use the application at the stage of development it currently is. Figure 1 shows what the user interface looks like.

When speaking about the wasted computation, it is easy to see that the average number of parsing attempts is quadratic in respect to the sentence length. But there are some cases in which the time increases dramatically: Some words (which can act in different syntactic roles) increase the number of potential valid parses. These performance characteristics are shown in the Figure 2.

Due to this fact we had to limit the length of the chunks to 5 to have the processing complete in reasonable time. The processing times of texts from around 32k web pages are shown in Figure 3.

The average time to process one text document is around 2 minutes. But as can be seen from the Figure 3, some documents take as much as 3 hours to process. A simple calculation gets us that to process our data we would need around 3 years of non-stop computing (on a single computer).

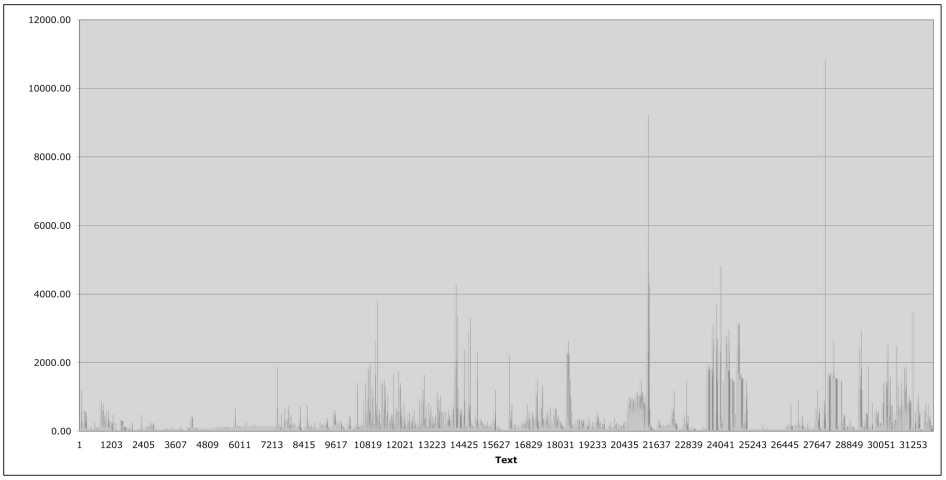

Figure 3. Text processing times

4. Using Grid

We did not dismiss the idea of processing all the data, though. If the processing takes 3 years on a single computer, it would take 10 days on 100 computers. And thanks to the BalticGrid project[4], we really had access to that many computers. Although the infrastructure is still in development, and there are some instabilities and interruptions, we still get a lot of computing power.

5. Results

After running the NLP application on the Web texts, we get a table like shown in Table 1.

The columns are:

(1) Job – many texts are sent for processing in a single Grid job. This number tells which job this file was processed in (so it is possible to re-process the job or look at log files if necessary).

[4] http://www.balticgrid.org

(2) File – each text is in its own file, and this column is the name of the file. It consists of two parts: the site where the text comes from, and the database id of the file (so it is possible to track down the source of the document in Web crawler database for additional information if needed).
(3) Words – the number of words in the document.
(4) Recognized – the percentage of words recognized by the NLP application.
(5) Chunks – the number of chunks recognized by the NLP application.
(6) Avg. size – the average size of a chunk.

Table 1. SemTi-Kamols application output

Job	File	Words	Recognized	Chunks	Avg.size
1	118.chat.lv!3319.html	706	54.11	202	1.02
1	118.chat.lv!5488.html	631	78.29	235	1.65
1	118.delfi.lv!102032.html	100	64	38	1.08
1	118.delfi.lv!1022486.html	669	83.41	222	2.1
1	118.delfi.lv!1024336.html	463	42.12	115	0.71
1	118.delfi.lv!1030216.html	1792	36.83	324	0.75
1	118.delfi.lv!109899.html	100	64	38	1.08

The question now is: do these numbers give us any information and can they be used to compare text quality? If we look at the largest documents (sorted by the number of words), we get many documents that are recognized quite poorly (Table 2).

Table 2. Results sorted by text size

Job	File	Words	Recognized	Chunks	Avg.size
107	audits.hei.lv!267170.html	36173	85.26	13404	1.96
555	isec.gov.lv!893208.html	28519	90.87	11163	2.11
46	ai1.mii.lu.lv!1765285.html	27142	77.46	10479	1.55
70	andre.veiksme.lv!961689.h	25316	9.1	2296	0.09
557	isec.gov.lv!925746.html	25207	91.44	9362	2.25
284	cs.kompakts.lv!58470.htm	25064	11.35	2701	0.12
107	audits.hei.lv!297115.html	21276	88.56	7390	2.26
555	isec.gov.lv!898322.html	19928	87.61	7217	2.12
580	kiss.id.lv!865370.html	19739	20.12	3654	0.22
514	home.lanet.lv!1613519.htr	19649	47.94	6697	0.67
46	ai1.mii.lu.lv!1761064.html	18895	73.84	7344	1.4

Best recognized texts on the other hand get us very small documents, which are of very little interest (Table 3).

So what we really have to look at is the number of correctly recognized chunks (or total recognized text which we can obtain from the total number of words and the recognized percentage; coincidentally, these numbers give very similar ordering for the documents):

Now we get some really interesting results. The top documents include:
(1) Financial law explanations.
(2) School program descriptions.
(3) Latvian folklore.
(4) Classical works of Latvian novelists.

Another interesting observation: if we look at the graph of top 200 entries from this list, we see that the quality (and document size) degrades very rapidly (Figure 4), and from around 32k documents (that's the most MS Excel® application can deal with

Table 3. Results sorted by recognized text percentage

Job	File	Words		Recognized	Chunks		Avg.size
39	abols.lanet.lv!122272.html		193	89.64		55	2.82
47	ai1.mii.lu.lv!1767524.html		124	83.87		31	2.81
4	159.148.57.120!587137.ht		436	88.99		124	2.78
4	159.148.57.120!841591.ht		210	88.57		60	2.75
549	isec.gov.lv!1074968.html		388	85.82		105	2.72
3	159.148.57.120!205895.ht		212	88.21		61	2.7
4	159.148.57.120!755839.ht		189	87.83		54	2.7
39	abols.lanet.lv!657218.html		143	88.81		42	2.69
36	80.232.169.5!301734.html		129	89.92		39	2.67
549	isec.gov.lv!1081389.html		369	86.45		105	2.63

if we want pretty graphs) only the top 17 (around %0.05) are interesting from the corpus point of view.

The documents following the top are of reasonable size and usually are either Web forum archives or single person works published on the web. The forums might be of big interest, but the problem in Latvia is that when it comes to writing online people

Table 4. Results sorted by recognized text size

Job	File	Words	Recognized %	Chunks	Avg.size	Recognized	
107	audits.hei.lv!267170.html	36173	85.26	13404	1.96	30841.10	Law comments/explanations
555	isec.gov.lv!893208.html	28519	90.87	11163	2.11	25915.22	Descriptions of school programs
557	isec.gov.lv!925746.html	25207	91.44	9362	2.25	23049.28	
46	ai1.mii.lu.lv!1765285.html	27142	77.46	10479	1.55	21024.19	Latvian proverbs
107	audits.hei.lv!297115.html	21276	88.56	7390	2.26	18842.03	
555	isec.gov.lv!898322.html	19928	87.61	7217	2.12	17458.92	
60	ailab.lv!82998.html	16965	87.59	7128	1.83	14859.64	R.Blaumanis - Purva bridējs
109	audits.hei.lv!429253.html	15464	90.27	5728	2.2	13959.35	
46	ai1.mii.lu.lv!1761064.html	18895	73.84	7344	1.4	13952.07	
59	ailab.lv!725485.html	15102	88.58	6606	1.79	13377.35	R.Blaumanis - Īstā līgaviņa
547	inventions.lza.lv!810755.html	12924	91.04	4904	2.18	11766.01	Latvian patent law
108	audits.hei.lv!365680.html	14416	79.34	4900	1.85	11437.65	
108	audits.hei.lv!326702.html	13058	87.37	4798	2.08	11408.77	
44	ai1.mii.lu.lv!1745453.html	15859	71.23	6140	1.31	11296.37	More proverbs (christmas theme)

use transliteration. This is mostly because of the poor support of Latvian character sets in applications in recent years. Now the situation is improving and Unicode support is available practically everywhere (but people are slow to change their habits).

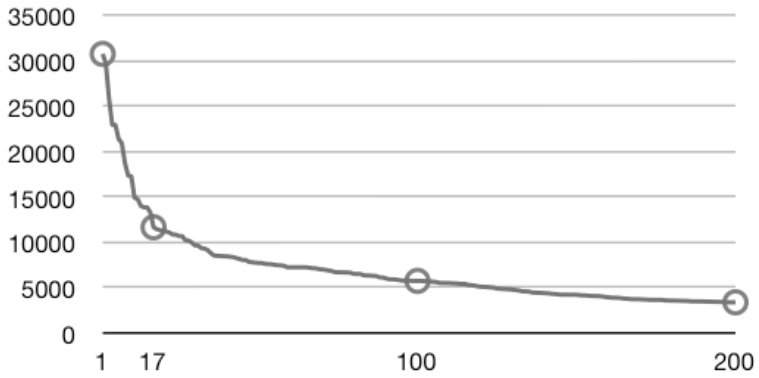

Figure 4. Text quality degradation

6. Conclusions

We have briefly looked at the problems of harvesting text corpora from the Web. The three components that allowed us to do this are:

(1) Text data from the web. We used texts from our own search engine, but today it might be reasonable to use Google API[5] for this purpose.
(2) NLP application. Practically any such application can be used if it can provide some numbers about the text syntactical and grammatical well-formedness.
(3) Lots of computing power. The processing time depends on the performance of the NLP application, but since there are lots of data then a single desktop computer will not be able to give any useful results in reasonable time. In our case we used Grid to deal with this problem.

The results are very promising: with our approach we were able to find sources of good quality text. With some effort it would be possible to do the text grouping into domain specific categories. Another possible work direction is to collect contemporary language texts, but in the case of Latvian language a tool to deal with transliteration must be developed first.

7. References

Bārzdiņs Guntis, Grūzītis Normunds, Nešpore Gunta and Saulīte Baiba, "Dependency-based hybrid model of syntactic analysis for the languages with a rather free word order", Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit (Eds.) *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007*, Tartu, May 2007. ISBN 978-9985-4-0514-7, 13–20

Bernardini Silvia, Baroni Marco and Evert Stefan, "A WaCky Introduction", Baroni, Marco and Bernardini, Silvia (eds.) 2006. *Wacky! Working papers on the Web as Corpus*. Bologna: GEDIT. [ISBN 88-6027-004-9], 9–40

Kilgarriff Adam, Grefenstette Gregory, "Introduction to the Special Issue on Web as Corpus", *Computational Linguistics 29 (3)*. 333–348

Jānis Džeriņš is a researcher in Institute of Mathematics and Computer Science, University of Latvia, participating in SemTi-Kamols and BalticGrid projects. He is now studying for PhD and the studies focus on knowledge representation and Semantic Web. He was also one of the main developers of the Latvian web search engine used in this paper. E-mail: janis.dzerins@lumii.lv.

Kristaps Džonsons is a researcher in the field of parallel/distributed computing. Recent projects include a system for explicit concurrency in Prolog and a high-throughput web crawler, designed specially for harvesting text data from the Web. Kristaps is also a strong figure in Latvian open source community, and has developed (among other things) a virtualisation tool sysjail for OpenBSD and NetBSD. E-mail: kristaps.dzonsons@latnet.lv.

[5] http://code.google.com/apis/