Feeding the "Brno Pipeline": The Case of Araneum Slovacum

Vladimír Benko[1,2]

[1] Slovak Academy of Sciences, Ľ. Štúr Institute of Linguistics
Panská 26, SK-81101 Bratislava, Slovakia
[2] Comenius University in Bratislava, UNESCO Chair in Plurilingual and Multicultural Communication
Šafárikovo nám. 6, SK-81499 Bratislava, Slovakia
vladob@juls.savba.sk
http://www.juls.savba.sk/~vladob

Abstract. Our paper is targeted at our experiences with a set of tools developed at the Masaryk University in Brno Faculty of Informatics, and it presents a case study describing our Autumn 2016 web crawl and its subsequent processing for the *Araneum Slovacum Maximum* web corpus.

Key words: web-based corpora, web data filtration and deduplication, Aranea Project

1 Introduction

The "Brno Pipeline" (BPL) is a term coined by Nikola Ljubešić [7] for a complete set of tools developed at Masaryk University in Brno, Faculty of Informatics that provide for the effective creation of corpora based on web-derived data. If complemented by an appropriate morphosyntactic tagger, BPL basically covers the whole process, practically without any need for additional programming capacity. Our work is a case study describing the use of BPL for the creation of *Araneum Slovacum Maximum* Slovak web corpus that is continuously being built since 2013 within the framework of the Aranea Project aimed at the creation of a family of comparable web corpora [3,4]. We will show some general considerations, describe the parameters of the whole process, and introduce some tools of our own implementing additional functionality not provided by BPL itself.

2 Data Crawling

The main BPL component is SpiderLing[3], a crawler highly optimized for downloading textual data from the web [12]. The tool also integrates modules

[3] http://corpus.tools/wiki/SpiderLing

Aleš Horák, Pavel Rychlý, Adam Rambousek (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2016, pp. 19–27, 2016.

for web page encoding detection (Chared[4]), boilerplate removal (jusText[5]; [8]), trigram-based language identification (trigrams[6]), and detection and removal of identical documents. The user-supplied input consists of a text sample for the respective language to be used to build a model for the language identification procedure, and a set of seed URL addresses needed for bootstrapping the crawl process.

Our experience manifests that the quality of the sample text is worth manual checking, as text fragments with non-standard orthography or those in foreign language(s) negatively influence the results of the language detection, resulting in large quantities of unwanted texts at the output of the procedure.

Within our Aranea Project, several methods of collecting URLs had been tested. The most convenient proved to be using the BootCaT[7] program [1] in the "seed words" mode. This may be quite simple if a PoS-tagged frequency word list is available. Such a list could be easily obtained from the Slovak National Corpus or, as in our case, from the already existing version of Araneum Slovacum. We decided to use the list of 1,000 most frequent adverbs, which is the word class with rather general meaning and almost no inflection. The list was used for extraction of randomly chosen groups of 20 words that were subsequently submitted as seeds for BootCaT. During each BootCaT run, 200 triples were created to be submitted as search expressions for the Bing[8] search engine requiring retrieval of the maximum count of 50 URLs during each search. Our typical BootCaT session consisted of 5 runs, theoretically yielding as many as 50,000 URLs. This count, however, usually dropped down to some 25,000 to 45,000 after removing duplicate URLs, as well as those pointing to wrong document types (such as PDFs that could not be processed by SpiderLing).

Probably the most important issue in using SpiderLing is its consumption of RAM – the author(s) seem to have expected that only very powerful server configurations would be utilized. As all necessary data structures storing information on the visited web pages, as well as on "wrong" URLs and duplicates, are kept in the main memory, the crawling process "freezes" when allocation of more RAM is not possible. In our case we could afford to dedicate for the long-time crawling only a machine with 16 GB of RAM, which usually led to freezing after some 80 hours of SpiderLing operation. The new crawling had to be started with a fresh set of seed URLs from scratch, risking the potentially high amount of duplicates appearing in the crawled data.

The RAM consumption, however, can be influenced by several user-settable parameters, with one of them being the restriction on top-level domain (TLD)

[4] http://corpus.tools/wiki/Chared

[5] http://corpus.tools/wiki/Justext

[6] http://code.activestate.com/recipes/326576-language-detection-using-character-trigrams/

[7] http://bootcat.sslmit.unibo.it/

[8] http://www.bing.com/

of the crawled documents. No restriction on TLDs results in an increase of RAM consumption, which may not be quite an intuitive behaviour.

During the SpiderLing operation, exact duplicates are identified (but not removed) on fly. Removal of the dupes can be performed at the end of a crawling session by a simple script. The resulting text file is in a "one line per paragraph" format containing light-weight XML markup describing docs, paragraphs and (optionally) the deleted boilerplate data.

3 Pre-Tokenization Filtration

Filtration aims at removing the documents containing texts that do not adhere to a predefined quality standard, and also those containing (partially) duplicate contents. This process is (at least in part) language-dependent, so we have written a series of filters that are being sequentially applied to the source data. Our filters typically consist of two components – the analyser generates a list of documents with a certain parameter above/under the specified threshold, and the removal procedure uses this list to split the input file into two parts, one containing the "good" docs and the other the "bad" ones. The advantage of such implementation is that the removal procedure can be universal, i.e., filter-independent, and also the fact that the removed documents can be subsequently analysed to provide for optimizing the parameters of filtration. Most of our own tools have been written in a rather "vintage" programming language, flex based on regular expressions and C code. The disadvantage of this approach is that flex is not compliant with Unicode (UTF-8), i.e., all computations have to be performed on the byte level only, and multi-byte UTF-8 characters must be treated by the programmer him/herself. On the other hand, flex programs tend to execute (at least) by order of magnitude faster than those written in an interpreted language such as Python and the actual speed of a filter is typically on par with plain file copying.

As we usually work with very large files, the sequence of filters can be conveniently optimized in order to remove most of the "wrong" data during the first step(s). The optimal sequence, however, is usually language-dependent, and in case of Slovak it is as follows:

1. Identification and removal of "insufficiently Slovak" documents. As the trigram language identification module is usually not able to cope with the differences among languages with similar character frequency distributions, not only lots of Czech texts appear in the data, but also some Croatian, Serbian, Slovene texts can be seen there. Our supplementary filter is based on counting the average frequencies of Slovak letters with diacritics, and separately counting two special cases: the missing "š/Š" a "ž/Ž" usually indicate an encoding issue (mostly Windows 1250 encoding misidentified as ISO 8859-2), while missing "ľ/Ľ" may mean that it is in fact a Czech text.

2. Identification and removal of exact duplicates by the fingerprint method [2]. These could not have been removed by SpiderLing itself, if sev-

eral independent crawling sessions had been performed. (The partial duplicates would be removed after tokenization only.)

3. Identification and removal of "too Czech" documents. Despite having passed the Slovak filter, some documents may nonetheless contain Czech text fragments. An algorithm analogous to that of Slovak is used, with the main difference being that the characters "ě/Ě", "ř/Ř" and "ů/Ů" (not present in Slovak) are counted.

4. Identification and removal of documents with incorrectly interpreted encoding, containing artefacts such as "po hrebeĹˆoch hĂ'r ÄŒeska, Slovenska a PoÄžska" (instead of "po hrebeňoch hôr Česka, Slovenska a Poľska") caused by treating a UTF-8 as 8-bit encoding. Quite often only a small fragment of a document may be affected. We, however, prefer removing such documents as a whole.

4 Tokenization

A standard BPL tool for tokenization is unitok [9], complemented by a language-specific parameter file. As no Slovak parameter file was present in the standard unitok distribution, we have created a new one based on the analogous Czech file. The new contents consist mainly of a list of period-final abbreviations, partially translated from the Czech list, and subsequently updated by abbreviations actually found in the Slovak corpus data. The only major tokenization policy change was the decision of tokenizing the "multi-period" abbreviations such as "s.r.o." as three separate tokens, so that it would be more compatible with the language model used by the tagger.

As the Python code takes much longer to execute than the flex filters, we usually run the tokenization as 4 processes in parallel to make use of the multicore processor of our server.

5 Post-Tokenization Filtration

Some encoding and other issues are easier to detect in an already tokenized text, as the regular expressions can rely on correct word boundaries. This is why some filters are better run after tokenization only. One of such filters is detecting situations defined as "uppercase letter with diacritics inside otherwise lowercase word". There may be several causes of this a phenomenon, such as a typo – incorrectly pressed SHIFT key ("vŠetko", "antikvariÁt"), "lost" spaces between two words ("vŽiline", "voŠvajciarsku"), incorrect interpretation of encoding ("veŸmi", "zÄava"), or even corrupted HTML entities ("nbspŠali"). We must, however, be cautious here – some of the tokens detected by this simplistic approach could represent legitimate neologisms with non-standard orthography ("eŠkola", "eČajovňa").

Table 1: Explanation of *ztag*.

0	The word has not been found in the lexicon, lemma has been just copied from word form.
1	The word form has been found in the lexicon and the lemma has been assigned unambiguously.
2, 3, 4	The word form has been found in the lexicon with 2 to 4-way ambiguity. Lemma contains all possible variants separated by a vertical line (" | ").

6 Detection of Near-Duplicate Documents

The next important processing step is deduplication. We can conveniently use another BPL component here – the Onion tool[9] [8]. The principle of its operation and testing its various settings was treated in our work [2]. We only mention two parameters here: deduplication is performed on 5-grams with similarity threshold level 0.9.

7 Morphosyntactic Annotation

The tagging process is mentioned only briefly here, as the tagger itself is neither part of BPL, nor our own tool, and also because annotation deserves a paper of its own. Besides the use of the tagger itself, our morphosyntactic annotation involves several additional steps: pre-tagging and post-tagging filters performing the "lemmatization" of punctuation and special graphic characters, marking the out-of-vocabulary (OOV) tokens and mapping the "native" tags universal PoS-only tags.

Our Slovak corpora are currently being tagged by TreeTagger[10] [11], using our own language model trained on the manually disambiguated *rmak-4.0* Slovak corpus [11] and the updated SNC morphological database using the SNC tagset [5].

TreeTagger also implements a guesser assigning tags to tokens not found in the lexicon. However, it does not try to guess lemmas for such tokens. In our Aranea corpora, the special attribute *ztag* is used to indicate the result of tagging, see Table 1 for explanation.

The tag attribute can be conveniently used in analysing the results of tagging, as problematic phenomena in the corpus can be queried explicitely.

The pre-tagging and post-tagging filter modify the lemmas and tags for punctuation and special graphic characters, providing what we may call a

[9] `http://corpus.tools/wiki/Onion`

[10] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

[11] `http://korpus.sk/ver_r(2d)mak.html`

Table 2: Resulting vertical file.

word	lemma	atag	tag	ztag
Dropbox	Dropbox	Nn	SSis4	0
i	i	Cj	O	1
SkyDrive	SkyDrive	Nn	SSns6	0
buď	budiť | byť	Vb	VMesb+	2
inštalujete	inštalovať	Vb	VKjpb+	1
,	,	Zz	Z	1
alebo	alebo	Cj	O	1
používate	používať	Vb	VKepb+	1
jeho	jeho	Pn	PUfs2	1
webového	webový	Aj	AAms2x	1
klienta	klient	Nn	SSms2	1
.	.	Zz	Z.	1

"lemmatization". For example, several Unicode representations of an apostrophe are retained as "word forms", but mapped to an "ASCII apostrophe" at the lemma level.

Similarly to other Aranea corpora, the "native" tags are mapped to Araneum Universal Tagset (AUT)[12], providing a parallel level of annotation. The respective values from the AUT tagset in the Aranea corpora are stored in the atag attribute.

The resulting vertical file after all annotation steps contains five attributes: *word, lemma, atag, tag* and *ztag*, see for example Table 2.

We can see two cases of an OOV item in our sentence, as well as a case of a 2-way ambiguous lemma – both variants, however, being incorrect in this particular case.

8 Paragraph-level Deduplication

Our corpus data can be utilized both as source data for various NLP projects or in a traditional way for "manual" analysis by means of a corpus manager. Depending on the mode of use, we implement two different policies of paragraph-level deduplication. For the NLP use, where nobody is expected to analyse the data by "reading" it, we prefer that the dupe paragraphs be deleted. For traditional work with a corpus manager we do not want to "destroy" the cohesion of the text by "randomly" deleting paragraphs inside a document. In this case we only mark the dupes so that they do not appear in the results of

[12] `http://aranea.juls.savba.sk/aranea_about/aut.html`

query operations, yet they could be displayed at the boundary of duplicate and non-duplicate text.

In both cases we use Onion with standard settings: 5-grams with similarity threshold 0.9.

9 Corpus Managers

The Araneum Slovacum is used in our Institute by lexicographers within our local installation of Sketch Engine[13] [6]. It is, however, available also for the general public at the NoSketch Engine[14] [10] Aranea Project portal.

10 The Autumn 2016 *Araneum Slovacum Maximum* Crawl

This section brings some data on our latest crawl and processing session performed in October 2016. The respective processing steps are shown in Table 3 and are accompanied by the relevant data on sizes and times. The crawling process itself consisting of six separate sessions is not included in the table. During this crawl we decided to experiment with releasing the TLD restriction.

As seen in Table 3, our decision not to limit the TLDs of the crawled web pages caused a large amount of "insufficiently Slovak" texts being removed by the very first filter. A brief checking reveals that most of the removed texts are Czech. They are not going to be disposed – we can use them during the next upgrade of our Czech *Araneum Bohemicum Corpus*. The bottom line, however, is that not setting TSD was probably not a good decision.

11 Conclusion and Further Work

After using the BPL tools for a fairly long time we can say that they represent a mature, efficient and robust set providing for all main procedures necessary to build a web corpus of our own. This also means that, having spare programming resources at hand, these can be targeted to language-dependent filtration and tiny improvements of the whole process. We can also see that implementation of the supplementary tools in flex tends to cut significantly the processing times, which is also the main reason why we have not decided yet to rewrite them into Python.

Our next plans – besides the fine-tuning of the whole process – includes testing some alternative tools, most notably the taggers.

Acknowledgments. The presented results were partially obtained under the VEGA Grant Agency Project No. 2/0015/14 (2014–2016).

[13] https://www.sketchengine.co.uk/
[14] https://nlp.fi.muni.cz/trac/noske

Table 3: Crawling results.

Operation	Output	Processing time (hh:mm)
Merging crawled text data from six SpiderLing sessions, assigning document Ids, fixing minor URL issues introduced by SpiderLing markup	7,108,601 docs 35.99 GB	n/a
Identifying and removing "insufficiently Slovak" documents	1,775,619 docs (75.02% removed) 9.41 GB	0:20
Identifying and removing exact duplicates by fingerprint method	1,370,075 docs (22.84% removed) 7.37 GB	0:17
Removing survived HTML markup and normalizing encoding (Unicode spaces, composite accents, soft hyphens, etc.)	7.36 GB	0:06
Removing successive duplicate paragraphs (by uniq)	7.31 GB	0:05
Identifying and removing "too Czech" documents	1,276,592 docs (6.82% removed) 6.51 GB	0:04
Identifying and removing documents with encoding issues	1,272,622 docs (0.31% removed) 6.49 GB	0:03
Tokenization by Unitok (4 parallel processes, custom Slovak parameter file)	980,058,957 tokens 7.39 GB	1:56
Truncating long tokens	7.39 GB	0:05
Identifying and removing documents with encoding issues (bis)	1,269,852 docs (0.22% removed) 7.36 GB	0:04
Segmenting to sentences (rudimentary rule-based algorithm)	56,969,058 sents 7.87 GB	0:06
Identifying and removing partially identical documents by Onion (5-grams, similarity threshold 0.9)	754,360 docs 559,387,978 tokens (42.63% removed) 4.50 GB	0:27
Pre-tagging filtration of punctuation and special graphic characters		0:03
Tagging by Tree Tagger with custom Slovak language model (4 parallel processes)	10.60 GB	0:56
Restoring original wordforms, marking the out-of-vocabulary (OOV) tokens (ztag), mapping native SNK tags to "PoS-only" AUT tagset (atag).	82,786,567 tokens marked OOV (6.43%)	0:08
Identifying and removing or marking partially duplicate paragraphs by Onion	(not performed yet at present)	0:0

References

1. Baroni, B., Bernardini, S.: BootCaT: Bootstrapping corpora and terms from the web. In: Proc. 4th Int. Conf. on Language Resources and Evaluation, Lisbon : ELRA (2004)
2. Benko, V: Data Deduplication in Slovak Corpora. In: Slovko 2013: Natural Language Processing, Corpus Linguistics, E-learning, pp. 27–39. RAM-Verlag, Lüdenscheid (2013)
3. Benko, V.: Aranea: Yet another Family of (Comparable) Web Corpora. In: Text, Speech, and Dialogue. 17th International Conference, TSD 2014 Brno, Czech Republic, September 8—12. Proceedings. Eds. P. Sojka et al. Cham – Heidelberg – New York – Dordrecht – London : Springer, pp. 21—29. ISBN 978-3-319-10816-2. (2014)
4. Benko, V.: Two Years of Aranea: Increasing Counts and Tuning the Pipeline. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC). Portorož : European Language Resources Association (2016) pp. 4245–4248. ISBN 978-2-9517408-9-1. (2016)
5. Garabík, R., Šimková, M.: Slovak Morphosyntactic Tagset. Journal of Language Modelling, 0(1), pp. 41–63 (2012)
6. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proc. XI EURALEX Int. Congress, Lorient, pp. 105–116 (2004)
7. Ljubešić, N., Klubička, F.: {bs,hr,sr}WaC – Web corpora of Bosnian, Croatian and Serbian. Proceedings of the 9th Web as Corpus Workshop (WaC-9). Gothenburg, Sweden. (2014)
8. Pomikálek, J.: Removing Boilerplate and Duplicate Content from Web Corpora. Ph.D. thesis, Masaryk University, Brno (2011)
9. Michelfeit, J., Pomkálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Ad-vances in Slavonic Natural Language Processing, RASLAN 2014, pp. 71—75, 2014. Brno: NLP Consulting (2014)
10. Rychlý, P.: Manatee/Bonito – A Modular Corpus Manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing. pp. 65–70. Masaryk University, Brno (2007)
11. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of International Conference on New Methods in Language Processing. Manchester (1994)
12. Suchomel V., Pomikálek J.: Efficient Web Crawling for Large Text Corpora. In: 7th Web as Corpus Workshop (WAC-7), Lyon, France, pp. 24–31. (2012)