Manual vs Automatic Bitext Extraction

Aibek Makazhanov, Bagdat Myrzakhmetov, Zhenisbek Assylbekov
Nazarbayev University, National Laboratory Astana
Nazarbayev University, School of Science and Technology
53 Kabanbay Batyr ave., Astana, Kazakhstan
aibek.makazhanov@nu.edu.kz, bagdat.myrzakhmetov@nu.edu.kz, zhassylbekov@nu.edu.kz

Abstract

We compare manual and automatic approaches to the problem of extracting bitexts from the Web in the framework of a case study on building a Russian-Kazakh parallel corpus. Our findings suggest that targeted, site-specific crawling results in cleaner bitexts with a higher ratio of parallel sentences. We also find that general crawlers combined with boilerplate removal tools tend to retrieve shorter texts, as some content gets cleaned out with the markup. When it comes to sentence splitting and alignment we show that investing some effort in data pre- and post-processing as well as fiddling with off-the-shelf solutions pays a noticeable dividend. Overall we observe that, depending on the source, automatic bitext extraction methods may lack severely in coverage (retrieve fewer sentence pairs) and on average are fewer precise (retrieve less *parallel* sentence pairs). We conclude that if one aims at extracting high-quality bitexts for a small number of language pairs, automatic methods best be avoided, or at least used with caution.

Keywords: bitext extraction, crawling, sentence alignment

1. Introduction

The Web has long been used as one of, if not, the most important source for obtaining language resources of various purpose, size and quality. Given the large amount and accessibility of text available in a variety of languages, it is particularly appealing to mine the Web for parallel and comparable corpora (bitexts), resources essential for building and evaluating data-driven machine translation systems.

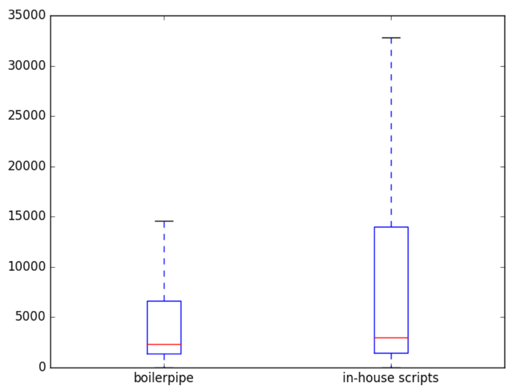
The task of extracting bitexts from the Web typically involves the following four major consecutive steps: (i) data collection (crawling), (ii) document alignment, (iii) sentence segmentation, and (iv) sentence alignment. Depending on the degree of user involvement in the first two steps, approaches to the problem can be classified roughly as: (i) manual methods (Koehn, 2005; Tiedemann, 2007; Myrzakhmetov et al., 2016) that crawl specific websites and usually rely on their idiosyncratic properties at the document alignment step; (ii) semi-automatic methods (Esplà-Gomis and Forcada, 2010; Papavassiliou et al., 2013) that make no site-specific assumptions, but still require a target list of URLs to extract bitexts from; (iii) completely automatic methods (Smith et al., 2013; Ljubešić et al., 2016; Resnik and Smith, 2003) that crawl the entire Web and produce whatever bitexts they can.

Choosing appropriate extraction strategy depends greatly on the task at hand. Thus, if the goal is to extract bitexts for dozens of language pairs automatic approaches may come in handy. However, if one needs a high quality parallel corpus for just one or two pairs of languages, what is the difference between bitexts obtained by manual and automatic methods? Is the final quality of manual extraction worth time and effort? To our disappointment, we could not find detailed answers to these questions in the literature. Various implementations of the extraction strategies are typically evaluated in terms of the alignment quality of the obtained bitexts and/or performance of SMT systems trained on those bitexts. Comparative evaluation mostly concerns different implementations of the semi-automatic approaches (Toral et al., 2014; Esplà-Gomis et al., 2014; Laranjeira et al., 2014). We believe that even a rough estimate of the quality/quantity trade off between bitexts obtained manually and automatically can provide important insights and guidelines for building parallel corpora. In this work we attempt to provide such an estimate.

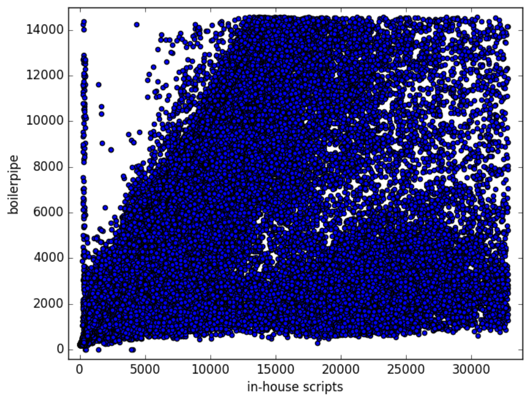
We divide the process of bitext extraction (BE) in two stages as a pre- and post-document alignment (DA) stage. The pre-DA stage involves crawling and DA itself. The post-DA stage is about sentence splitting and alignment, with possible cleaning steps that may improve the quality of an extracted bitext. We compare (semi-) automatic and manual approaches to both BE stages using a range of metrics and manual and automatic estimates.

1.1. General Setting

Before proceeding let us describe a particular setting considered herein. First, the present work relies heavily on particular sources, namely certain websites that adhere to the same language policy of making multilingual releases in Kazakh and Russian languages. Such websites almost always provide page-level alignment, e.g. a Russian version of a page contains a direct link to a Kazakh version of the same page and vice versa. Thus, we handle document alignment at the stage of crawling. Second, for BE we use only freely accessible tools; hence our conclusions may not generalize to commercial analogs, if any. Lastly, to conduct our experiments we use the following four websites as the development set: `adilet.zan.kz`, `akorda.kz`, `astana.gov.kz`, and `strategy2050.kz`, which are shortened to `adilet`, `akorda`, `astana`, and `strategy`, and hereinafter collectively referred to as 'four (web)sites'.



(a) Side-by-side boxplots



(b) Scatter plot

Figure 1: Lengths (in characters) of the texts extracted by the boilerpipe and in-house scripts

2. General vs Targeted Crawling

The crawling step can be divided into three parts: (1) obtaining the links to parallel web pages (spidering), (2) downloading raw HTML files, (3) extracting plain texts from the downloaded web pages. The downloading step is trivial; it is steps (1) and (3) where general and targeted crawling diverge.

General crawling. For a given page-aligned website we obtain the links to all pages in one language (*source* language), using the `GNU wget` tool in spider mode and specifying a regular expression pattern to accept only the links to pages in *source* language (e.g. `'/ru/'` for the links in Russian). Then we use `GNU wget` again to download all the pages in the source language. Next from raw HTML source pages we obtain the corresponding links to pages in *target* language from the raw source language HTML files and download all the web pages in the target language. Finally, we remove boiler-plate text, i.e. navigational elements, templates, and advertisements which are not related to the main content, and extract plain text from the downloaded HTML-files using the `boilerpipe` tool. We also remove duplicates using `fdupes` tool[1].

Targeted crawling. For this task we use site-specific crawlers, which are essentially in-house python scripts (one per website). Here instead of obtaining the links and later using them to download the content. We can traverse certain branches of a website (say, news), download pages in both source and target languages, and extract meaningful content relying on the HTML-structure of the site: all in one go.

2.1. Comparison of the Spider Methods

For a given website, let B be the set of unique source-language links obtained by the baseline method, and I be the set of unique source-language links obtained by the in-house scripts. Cardinalities of B, I, $B \cap I$, $B \setminus I$, $I \setminus B$ for the four websites are given in Table 1. One should notice that the numbers in Table 1 are not directly comparable between the two methods: in the baseline approach we use the `wget` tool to obtain *all* the URLs from a given website, whereas the hand-made scripts are designed to crawl only particular branches (legal documents, news, announcements, etc.) on each website. Theoretically, `wget` in `--spider` mode should obtain *all* links from a given website, and one should expect that $B \supset I$ (or almost so depending on a time lag between two crawls), but this is not always the case as it can be seen from Table 1. Moreover, in some cases the in house scripts obtained more links, and as our closer analysis reveals in the case of `akorda` there were 3 unique links per page in B.

2.2. Comparison of the Text Extraction Methods

We consider the links from $B \cap I$, i.e. those links which were obtained by both the baseline and the in-house methods. According to Table 1, there are 108,960 of such links per each language. We downloaded the web pages from those links, and then we applied the `boilerpipe` tool and the in-house Python scripts to extract texts from them. After that character lengths were calculated for all texts, and outliers were removed using the Tukey's IQR rule (Tukey, 1977).

Side-by-side boxplots (Fig. 1a) show that the automatic extractors seem to produce longer texts than the automatic and site-agnostic `boilerpipe`. The scatter plot (Fig. 1b) provides more detail clearly showing that the majority of longer texts were extracted by the in-house scripts. Average length of the extracted texts is 3,226.56 for `boilerpipe` and 5,290.62 for the in-house scripts. According to the Shapiro-Wilk test (Shapiro and Wilk, 1965) normality is violated for text lengths in both cases, $p < 0.001$. Wilcoxon signed-rank test (Wilcoxon, 1945), confirms the hypothesis that `boilerpipe` produces shorter texts on average, $p < 0.001$. Thus, boilerpipe at best provides comparable quality of text extraction, and in worst case it throws away useful chunks of text (main content).

[1]https://github.com/adrianlopezroche/fdupes

Website	$\lvert B\rvert$	$\lvert I\rvert$	$\lvert B\cap I\rvert$	$\lvert B\setminus I\rvert$	$\lvert I\setminus B\rvert$
`adilet`	86,234	87,363	79,088	7,146	8,275
`akorda`	33,572	6,357	6,347	27,225	10
`astana`	28,067	6,512	6,496	21,571	16
`strategy`	28,078	28,253	17,029	9,246	11,224
TOTAL	175,951	128,485	108,960	65188	19525

Table 1: Number of links obtained by the baseline (B) and by the in-house (I) methods

Website	Annotator-1		Annotator-2		Annotator-3		Final	
	Wget + boilerp.	In-house scripts	Wget + boilerp.	In-house scripts	Wget + boilerp.	In-house scripts	Wget + boilerp.	In-house scripts
`adilet`	0.7950	0.9375	0.8050	0.9425	0.7125	0.8825	0.7500	0.9075
`akorda`	0.8200	0.9525	0.8125	0.9450	0.7625	0.8675	0.7750	0.9050
`astana`	0.6400	0.7925	0.6350	0.7950	0.5050	0.7325	0.5975	0.7400
`strategy`	0.7550	0.7700	0.7200	0.7525	0.6250	0.6575	0.6425	0.6900

Table 2: Crawling vs Alignment: manual evaluation, proportions of parallel pairs are provided

Metrics	`Wget+boilerpipe`				In-house crawling scripts			
	adilet	akorda	astana	strategy	adilet	akorda	astana	strategy
# parallel	6,522,758	94,855	208,515	250,536	22,728,878	70,949	63,729	201,671
parallel/total	0.7680	0.8309	0.5698	0.6680	0.8803	0.9116	0.7215	0.6650
short/parallel	0.3079	0.1211	0.0676	0.2276	0.6312	0.0190	0.0098	0.0202
junk/total	0.1260	0.0998	0.3897	0.1191	0.0743	0.0294	0.0610	0.0982

Table 3: Crawling vs Alignment: automatic evaluation

2.3. Impact of Crawling

To see how the two different ways of crawling affect the final quality of the bitext we applied both methods to the four websites and obtained two sets of parallel documents, which we proceed to split into sentences using NLTK Punkt tokenizer(Kiss and Strunk, 2006), and align with Hunalign(Varga et al., 2007). Thus we end up with two sets of bitexts which differ only in the way they were crawled. To compare these set we randomly sampled 400 pairs from each population[2] and asked three annotators if each pair of sentences intended to provide the same content in the two different languages[3]. The fourth annotator made the final decision. One can see from Table 2 that despite the differences in annotations, crawling with the in-house Python scripts consistently outperforms the `Wget+boilerpipe`-based crawling across the annotations. The difference in sample proportions for the 'Final' column is significant at 0.001 level for `adilet`, `akorda`, `astana`, and at 0.1 level for `strategy`, two-sided z-test for proportions. This supports the hypothesis that the in-house scripts result in *cleaner* bitexts than the `wget+boilerpipe`-based crawling.

[2]According to Cohran's sample size formula (Cochran, 2007), this is enough to ensure 5% margin of error at 95% confidence level.

[3]The rationale behind asking this question instead of the question "Are these sentences translations of each other?" is provided by (Resnik and Smith, 2003).

We can also compare the levels of noise in the bitexts by considering the following metrics: (i) proportion of short pairs among parallel pairs; (ii) proportion of obviously nonparallel pairs (junk) among all pairs. Here a pair considered short whenever the lengths of each sentence in a pair does not exceed *three* words. Junk pairs are defined as follows: (i) at least one of the sides (source or target) is empty; (ii) at least one of the sides does not contain any letters (Latin and Cyrillic); (iii) both sides are identical after tokenization and lowercasing. The results of this comparison across the four websites are given in Table 3. We can see again that the targeted crawling produces cleaner bitexts than that of general.

2.4. Automatic Evaluation

Although manual evaluation is the best way to estimate alignment quality, it requires significant time and effort. Therefore we follow Assylbekov et al. (2016) and use the set of length- and context-based features to automatically estimate alignment quality for a given pair of sentences. We train a number of classifiers on the data obtained from the previous experiment, i.e. manually annotated 3200 sentence pairs (8 samples 400 sentence pairs each). From a number of tested classifiers we chose the gradient boosting classifier since it achieved the highest accuracy and low variance.
The results of the automatic evaluation on the entire data set are given in Table 3. One can see that ratio of parallel pairs are within 5% margin of error from

Metrics	Manual BE pipeline				Bitextor			
	adilet	akorda	astana	strategy	adilet	akorda	astana	strategy
# of parallel	22,658,881	70,025	70,449	202,633	296,650	42,217	20,129	67,643
parallel/total	0.9075	0.9484	0.7992	0.7303	0.8725	0.7900	0.3000	0.4350
short/parallel	0.3412	0.0095	0.0075	0.0195	0.0000	0.0285	0.0250	0.0057
junk/total	0.0000	0.0000	0.0000	0.0000	0.0000	0.0025	0.0000	0.0000

Table 4: End-to-end bitext extraction: manual vs automatic

Method	P	P/T	S/P	J/T
Automatic SSA	5,766,307	0.7946	0.1701	0.0657
Domain-adapted SS	5,738,912	0.8068	0,1765	0,0650
SA + AD	5,748,352	0.8052	0,1771	0,0649
SA + L + MD	5,750,528	0.8436	0,1742	0,0264
SA + L + MD + JR	5,750,497	0.8446	0,0944	0,0000

Table 5: Evaluation of consecutive modification of SSA tools: P - # of sentence pairs deemed parallel; T - total pairs; S - short pairs; J - junk pairs; SS - sentence splitting; SA - sentence alignment; AD - automatic dictionary; L - lemmatization; MD - manual dictionary; JR - junk removal.

the estimates in the 'Final' column of Table 2. Therefore we have a quick and reliable method to get the approximate proportion (and number) of parallel sentence pairs in a given corpus, which can be used to evaluate any further modifications to the BE pipeline.

3. Bitext Extraction Beyond Crawling

Once crawling and document alignment is complete, we can proceed to the next stage of BE, which includes sentence splitting and alignment (SSA). Here we also distinguish between automatic and manual approaches. An automatic approach entails direct application of the SSA tools, whereas a manual approach assumes pre- and post-processing of data, as well as adapting SSA tools for a particular language pair.

Starting with the parallel documents that we have extracted at the crawling stage, we perform SSA: first automatically and then with successive application of the following modifications: (i) domain adaptation of a sentence splitter (retraining Punkt tokenizer); (ii) using automatically extracted dictionary for sentence alignment (`-realign` option of Hunalign); (iii) using hand-crafted dictionary in conjunction with preliminary lemmatization of texts[4]; (iv) removal of junk pairs (cf. subsection 2.3.).

After consecutive application of all the modifications we have compared the resulting bitext with the one that was extracted by a popular off-the-shelf tool `Bitextor` (Esplà-Gomis and Forcada, 2010). Bitextor was run on the four sites on default settings. As it can be seen from Table 4 the manual pipeline extracts more parallel sentences (especially for `adilet` data) and is more accurate than an unsupervised automatic tool. However, on average, Bitextor produces cleaner bitexts with lower short/parallel ratios and surprisingly low amount of junk.

Lastly we would like to assess the dynamics of modifying SSA tools. To this end, we perform automatic evaluation of the bitexts extracted after successive application of each modification. The results of this assessment (averaged over the four sites) are given in Table5. As it can be seen consecutive application of the modifications "cleanses" resulting bitexts, as the ratio of junk pairs decreases. At the same time there is a steady increase in the ratio of parallel sentences. However, the number of parallel and short/parallel pairs do not change monotonically. When we apply domain adapted SS the amount of parallel pairs drops almost 30K, but after that enjoys a steady growth. It turns out that as we provided Punkt sentence splitter with the list of abbreviations a total number of sentences dropped as the number of incorrect segmentations (due to dotted abbreviations) has been reduced.

4. Related Work

Koehn (2005) employs manual approach by mining the Euro Parliament Proceedings. The documents are aligned by comparing time stamps and the HTML structure. Similarly Tiedemann (2007) focuses on the single website to obtain movie subtitles and align those using time stamps. Smith et al. (2013) use the CommonCrawl Web snapshot. Candidate documents are retrieved by matching language names and codes to URLs. The candidates are later aligned using an extension of the STRAND algorithm (Resnik and Smith, 2003). Ljubešić et al. (2016) employ a combination of the SpiderLing crawler and the Bitextor. The former crawls the top level domains and saves the links to the websites whose pages are identified as written in predefined target languages. This list of links is later provided to the Bitextor, which aligns documents using their various properties, such as size, file name, difference in length etc.

[4]For lemmatization we use: Mystem (Segalovich, 2003) for Russian, and a data-driven morphological disambiguator (Makhambetov et al., 2015) for Kazakh.

5. Conclusion

We have compared manual and automatic approaches to the problem of extracting bitexts from the Web in the framework of a case study on building a Russian-Kazakh parallel corpus. We conclude that if one aims at extracting high quality bitexts for a small number of language pairs, automatic methods best be avoided, or at least used with caution. In the future we plan to expand the work on manual bitext extraction, and experiment with more Kazakhstani websites, including *not* page-aligned ones. Our ultimate goal is to build a high quality, decent-sized parallel corpus for the Russian-Kazakh pair.

6. Acknowledgements

This work has been funded by the Nazarbayev University under the research grant №129-2017/022-2017 and by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan under the research grant AP05134272.

7. Bibliographical References

Assylbekov, Z., Myrzakhmetov, B., and Makazhanov, A. (2016). Experiments with Russian to Kazakh Sentence Alignment. *Izvestija KGTU im.I.Razzakova*, 38(2):18–23.

Cochran, W. G. (2007). *Sampling techniques.* John Wiley & Sons.

Esplà-Gomis, M. and Forcada, M. (2010). Combining content-based and url-based heuristics to harvest aligned bitexts from multilingual sites with bitextor. *The Prague Bulletin of Mathematical Linguistics*, 93:77–86.

Esplà-Gomis, M., Klubička, F., Ljubešić, N., Ortiz-Rojas, S., Papavassiliou, V., and Prokopidis, P. (2014). Comparing two acquisition systems for automatically building an english-croatian parallel corpus from multilingual websites. In *LREC*, pages 1252–1258.

Kiss, T. and Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Laranjeira, B., Moreira, V. P., Villavicencio, A., Ramisch, C., and Finatto, M. J. B. (2014). Comparing the quality of focused crawlers and of the translation resources obtained from them. In *LREC*, pages 3572–3578. Citeseer.

Ljubešić, N., Esplà-Gomis, M., Toral, A., Rojas, S. O., and Klubička, F. (2016). Producing monolingual and parallel web corpora at the same time - spiderling and bitextor's love affair. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Makhambetov, O., Makazhanov, A., Sabyrgaliyev, I., and Yessenbayev, Z. (2015). Data-driven morphological analysis and disambiguation for kazakh. In *Proceedings of the 2015 Computational Linguistics and Intelligent Text Processing, Part I*, pages 151–163, Cairo, Egypt. Springer International Publishing.

Myrzakhmetov, B., Sultangazina, A., and Makazhanov, A. (2016). Identification of the parallel documents from multilingual news websites. In *Application of Information and Communication Technologies (AICT), 2016 IEEE 10th International Conference on*, pages 197–202. IEEE.

Papavassiliou, V., Prokopidis, P., and Thurmair, G. (2013). A modular open-source focused crawler for mining monolingual and bilingual corpora from the web. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 43–51.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Segalovich, I. (2003). A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *MLMTA*.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A. (2013). Dirt cheap web-scale parallel text from the common crawl. In *ACL (1)*, pages 1374–1383.

Tiedemann, J. (2007). Building a multilingual parallel subtitle corpus. *Proc. CLIN*, page 14.

Toral, A., Rubino, R., Esplà-Gomis, M., Pirinen, T., Way, A., and Ramirez-Sanchez, G. (2014). Extrinsic evaluation of web-crawlers in machine translation: a case study on croatian–english for the tourism domain. In *Proceedings of the 17th Conference of the European Association for Machine Translation (EAMT)*, pages 221–224.

Tukey, J. W. (1977). Exploratory data analysis.

Varga, D., Halacsy, P., Kornai, A., Nagy, V., Nemeth, L., and Tron, V. (2007). Parallel corpora for medium density languages. *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP 2005*, 292:247.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.