KazNLP: A Pipeline for Automated Processing of Texts Written in Kazakh Language

Zhandos Yessenbayev(✉), Zhanibek Kozhirbayev, and Aibek Makazhanov

National Laboratory Astana, Nur-Sultan, Kazakhstan
{zhyessenbayev,zhanibek.kozhirbayev,aibek.makazhanov}@nu.edu.kz

Abstract. We present the current results of our ongoing work on develop-ing tools and algorithms for processing Kazakh language in the framework of KazNLP project. The project is motivated by the need in accessible, easy to use, cross-platform, and well-documented automated text processing tools for Kazakh, particularly user generated text, which includes transliteration, code switching, and other artifacts of language-specific raw data that needs pre-processing. Thus, apart from a basic tokenization-tagging-parsing pipeline, and downstream applications such as named entity recognition and spell checking, KazNLP offers pre-processing tools such as text normalization and language identification. All of the KazNLP tools are released under the Creative Commons license. Since the detailed description of the methods and algorithms that were used in KazNLP are published or to be published in various venues, reference to which is given in the corresponding sections, this work provides just an overview of the tools and their performance level.

Keywords: Kazakh language · Natural language processing · Corpus linguistics · Computational linguistics · Programming tools

1 Introduction

Kazakh language is the official language of the Republic of Kazakhstan, spoken by over 12 million people. From the NLP point of view, Kazakh is an interesting research object that presents several challenges as an agglutinative language with complex morphology and a relatively free word order. However, NLP research on this language is rather scarce, while demand in automated processing of Kazakh text is rising. For example, consider automatic spellchecking, one of the classic NLP tasks. To our knowledge there are only two research papers [12,20] that discuss spell-checking for Kazakh. Moreover, the method described in [12] achieved only 83% accuracy, which was enough to outperform MS Word spell-checking system. Thus, while Kazakh spellcheckers are far from being perfect, there is almost no research in this direction. This is not surprising, because to achieve high performance on many NLP problems for languages with complex

A. Karpov and R. Potapova (Eds.): SPECOM 2020, LNAI 12335, pp. 657–666, 2020.
https://doi.org/10.1007/978-3-030-60276-5_63

morphology, one needs the very basic tools such as morphological analyzers and taggers. To be fair, we admit that such tools exist for Kazakh, but as far as we can tell, they are limited in access [7], ease of use [1,31], or the degree of readiness ("raw" experimental systems) [16]. Moreover, existing tools are documented either poorly or not at all and have not been evaluated on user generated text, such as internet reviews and message boards.

To address these issues, we have developed KazNLP, an open source Python library for processing Kazakh texts. KazNLP consists of the following modules, that can be used in isolation or in a pipeline: 1) initial normalization tool; 2) sentence-word tokenizer; 3) language identification tool; 4) morphological analyzer; 5) morphological tagger; 6) syntactic parser; 7) spelling checking and correction tool; 8) secondary normalization tool; 9) named entity recognizer. The set of tools is chosen to strike a balance between research and engineering applications, and will, hopefully, be extended in the future. To facilitate efficiency and performance trade-off, where possible, we have developed both statistical and neural implementations. Importantly, we avoided any hard-coded rules, to make the tools adaptable to the announced shift of the Kazakh alphabet from Cyrillic to Latin. The code is released under the *CC-SA-BY* license (Creative Commons Attribution 4.0), which allows any use, including commercial.

In what follows, we describe KazNLP in more detail. Specifically, Sect. 2 describes the datasets and corpora used in the development of KazNLP. Section 3 provides an overview of KazNLP components and corresponding NLP tasks. Section 4 briefly describes the software package and the demo web service. Finally, we draw conclusions and discuss future work in Sect. 5.

2 Text Datasets and Corpora

The development of KazNLP would not be possible without language resources described below. All the datasets are available either online or upon request.

2.1 Kazakh Language Corpus

The Kazakh Language Corpus (KLC) [17] was collected to aid linguistics, computational linguistics, and NLP research for Kazakh. It contains more than 135 million words in more than 400 thousand documents classified by genres into the following five sections: 1) *literary*; 2) *official*; 3) *scientific*; 4) *publicistic*; 5) *informal language*. KLC also has a portion of data annotated for syntax and morphology. It should be noted that initially the syntactic tagset comprised a compact set of syntactic categories, which were later improved during the development of Kazakh Dependency Treebank [13] described next.

2.2 Kazakh Dependency Treebank

Another important linguistic resource that lays the grounds of the current work is the Kazakh Dependency Treebank [13,14]. Following the best practices of the

Universal Dependencies (UD) guidelines [23] for consistent annotation of grammar, we have relabeled a part of KLC with lexical, morphological, and syntactic annotation that can be used by computer scientists working on language processing problems and by linguists likewise. The treebank contains about 61 K sentences and 934.7 K tokens (of those 772.8 K alphanumeric), stored in the UD-native CoNLL-U format. In addition, we annotated all the proper nouns of the corpus with such labels as a person's name, location, organization and others.

2.3 Other Datasets

To perform experiments on text normalization, language identification, and senti-ment analysis, we used a data set of user generated text collected from the three of the most popular Kazakhstani online news outlets, namely `nur.kz`, `zakon.kz`, and `tengrinews.kz` [21]. The noisy data was corrected and labeled semi-automatically resulting in total of 27,236 comments annotated for language and sentiment. Positive and negative comments amounted to 5995 (22%) and 7409 (27.2%), respectively, with the rest being neutral. In terms of languages, data set contains 63% of Kazakh texts, 34.4% of Russian texts and the rest is mixed comments (i.e. code-switched between Kazakh and Russian).

3 KazNLP Components

3.1 Initial Normalization Module

User generated content (UGC) generally refers to any type of content created by Internet users. UGC as a text is notoriously difficult to process due to prompt introduction of neologisms, peculiar spelling, code-switching or transliteration. All of this increases lexical variety, thereby aggravating the most prominent problems of NLP, such as out-of-vocabulary lexica and data sparseness. It has been shown [6] that certain preprocessing, known as lexical normalization or simply normalization, is required for them to work properly. Kazakhstani segment of Internet is not except from noisy UGC and the following cases are the usual suspects in wreaking the "spelling mayhem":

- *spontaneous transliteration*, e.g. Kazakh word "" can be spelled in three additional ways: "", "", and "biz";
- use of *homoglyphs*, e.g. Cyrillic letter "" (U+0456) can be replaced with Latin homoglyph "i" (U+0069);
- *code switching* – use of Russian words and expressions in Kazakh text and vice versa;
- *word transformations*, e.g. "","" instead of "" (great), or seg-mentation of words, e.g. " "or "——".

We have implemented a module for initial normalization of UGC. However, unlike with lexical normalization [6], we do not attempt to detect ill-formed words and recover their standard spelling. All that we really care about at this point is to provide an intermediate representation of the input UGC that will not necessarily match its lexically normalized version, but will be less sparse. Thus, we aim at improving performance of downstream applications by reducing vocabulary size (effectively, parameter space) and OOV rate. To this end, initial normalization does two things: (i) converts the input into a common script; (ii) recovers word transformations and does various minor replacements. Our simple *rule-based* approach (using regular expressions and dictionary) amounts to successive application of three straightforward procedures: (i) homoglyph resolution - substitutes Latin letters with Cyrillic counterparts (ii) common script transliteration - substitutes similar Latin and Cyrillic letters with common letter (iii) replacement and transformation - applies regular expressions to correct ill-formed word forms. It is difficult to perform a direct evaluation of this algorithm, but an extrinsic evaluation can be found in our previous work [21].

3.2 Sentence-Word Tokenizer

Sentence segmentation is a problem of segmenting text into sentences for further processing; and tokenization is a problem of segmenting text into chunks that for a certain task constitute basic units of operation (e.g. words, digits, etc.). At first glance, the problems might seem trivial, but this is not always the case, as many standard punctuation symbols might be used in different contexts (abbreviations, emoji, etc.), and the meaning of "tokens" and "sentences" may vary depending on a task at hand. Thus, to solve sentence and token segmentation problems one cannot blindly segment texts at the occurrences of certain symbols, and has to resort to a more sophisticated approach.

We are aware of ready to use tools that can be adapted to Kazakh, such as Punkt in NLTK [3], Elephant [4], and Apache OpenNLP [26]. However, they mostly use hand-crafted features. The only free-distributed tokenizer for Kazakh is based on the finite state transducer and is built into the morphological analyzer [31], which is not always convenient and necessary. Therefore, we decided to implement our own module for sentence splitting and word tokenization, which treats the problem as a single sequence labeling task.

The current version of KazNLP includes only HMM-based implementation of the module, but we have also experimented with an LSTM-based approach. This approach utilizes character embeddings, i.e. represent characters as vectors in a multidimensional continuous space. We evaluated our method on three typologically distant languages, namely Kazakh, English, and Italian. Experi-ments show that the performance on F1-measure achieves 95.60% on sentence and 99.61% word segmentation, respectively, for Kazakh language which is better than that of popular systems like Punkt and the Elephant. An interested reader is referred to our work in [30].

3.3 Language Identification Module

Language identification (LID) is the task of determining a language in which a given piece of information is presented be it text, audio or video. Early works on LID for texts have achieved near perfect accuracy when applied to small sets of monolingual texts, languages and domains, which led to a popular misconception that LID task was solved. The authors of [2] argue that LID is far from being solved, and show that as number of target languages grows and documents become shorter, performance of the standard methods drops. Another challenge that the authors mention, but do not address, is the fact that input text may be multilingual. Indeed, the results of the *Workshop on Computational Approaches to Code Switching* show that word-level LID on multilingual input is significantly harder than classical LID, i.e. document-level on monolingual input.

In KazNLP, we aim to provide tools for processing real world data, including noisy UGC, i.e. tweets, comments, Internet forum dialogs, etc. Apart from noise, there is a certain amount of Kazakh-Russian code-switching, which allows us to experiment with multilingual word-level LID. Following a common strategy of dealing with noisy UGC, prior to LID, we perform normalization as described above. For LID, we trained and applied character-based LSTM neural networks. The best performance of 99.73% was achieved for 150-character long sequences. Our results suggest improvement over the state-of-the-art for Kazakh language based on popular LangID [11] or Bayesian approach [9].

3.4 Morphological Analyzer and Tagger

Morphological analysis is the task of identifying the constituents of a word (root and morphemes), while morphological tagging is the task of identifying the most suitable analysis in the context of a sentence. Thus, this problems are closely related. A common approach for the agglutinative languages like Kazakh is to develop a finite state transducers to identify all possible morphological parses of a word [31], and on top of that apply a morphological disambiguation model based on statistical or other machine learning approaches [16,29].

In this module, we addressed the morphological analysis and tagging problem as a single sequence-to-sequence modeling task similar to neural machine transla-tion. In particular, we applied a character-level encoder-decoder neural models with attention using bi-directional LSTM network implemented in Open-NMT [8]. For comparison, we experimented with language independent packages like UDPi-pe [27] and Lemming [19]. In both tasks, our model outperformed the latter models, achieving the accuracy on morphological analysis of 87.8% and on mor-phological tagging of 96.8%.

3.5 Syntactic Parser

In computational linguistics and NLP, syntactic parsing usually amounts to determining a parse tree of a given sentence, according to a predefined grammar formalism, e.g. constituency, dependency, etc. grammars. In KazNLP, we adhere to dependency formalism and develop a dependency parser. Commonly used approaches to tackle this problem fall into two main classes: graph-based (MST-Parser [18]) and transition-based (Malt-parser [24]) parsing models. As an initial version, we have implemented a graph-based parser using data-driven statistical approach to compute weights of the search graph [32]. Thus, the goal is to find a minimum spanning tree in the given weighted directed graph. The performance of the parser in terms of unlabeled attachment score was 61.08, which is barely comparable to the state-of-the-art neural-based models. Nonetheless, the tool can be used in the domains with limited amount of data, where the advanced models are not applicable.

3.6 Spelling Correction Module

The spelling correction can be divided into two tasks: word recognition and error correction. For languages with a fairly straightforward morphology recognition may be reduced to a trivial dictionary look up. Correction is done through generating a list of possible suggestions: usually words within some minimal edit distance to a misspelled word. For agglutinative languages, such as Kazakh, even recognizing misspelled words becomes challenging as a single root may produce hundreds of word forms. It is practically infeasible to construct a dictionary with all possible word forms included: apart from being gigantic such a dictionary would be all but verifiable. For the same reason the correction task becomes challenging as well.

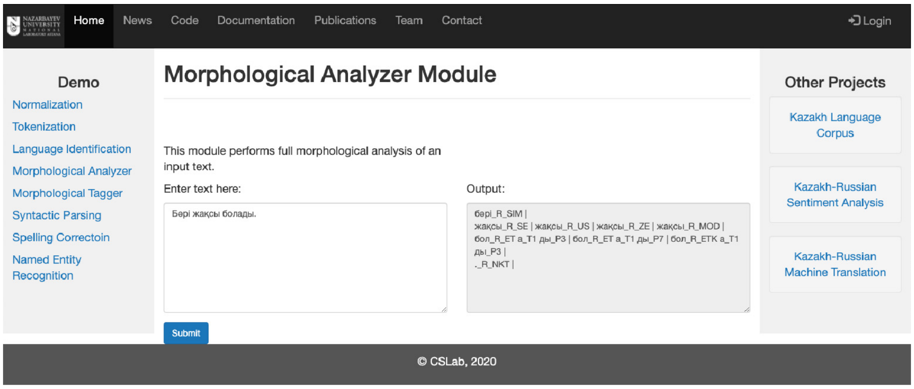
To address this problem, we followed the approach presented by Oflazer and Güzey [25]. In particular, we used a mixture of lexicon-based and generative approach by keeping a lexicon of roots and generating word forms from that lexicon on the fly for lookup and correct suggestions. To rank a list of suggestions, we use a Bayesian argument that combines error and source models. For our error model we employ a noisy channel-based approach proposed by Church and Gale [6]. Our source model is built upon the theoretical aspects that were used for morphological disambiguation in [5]. For the purpose of comparison we experimented with built-in Kazakh spelling packages in OpenOffice and Microsoft Office 2010. We show that although our method is more accurate than the open source and commercial analogues, achieving the overall accuracy of 83% in generating correct suggestions, the generation of those suggestions still needs improvement in terms of pruning and better ranking [12].

3.7 Secondary Normalization Module

In addition to the initial normalization and spelling correction modules, we implemented another tool to normalize and correct texts - secondary normalization module. The secondary normalization is designed to directly convert UGC to the standard language overcoming the drawbacks of two former stages, namely, incompleteness of rule-based approach. In this task, we applied a statistical machine translation strategy to convert from noisy texts (source language) to lexically correct texts (target language) given the corresponding parallel data set. The performance of this approach was 21,67 in terms of BLEU metrics, which is considered as a moderate result [22]. It should be noted, that this tool still can be used after initial normalization and spelling correction, as it also able to correctly deal with phrases.

3.8 Named Entity Recognizer

Named entity recognition (NER) is considered one of the important NLP task. It is a problem of recognizing real world objects found in a sentence, such as geographical location, person's name, organization, and others. There are several approaches based on manually created grammar rules, statistical models, and machine learning to solve the NER problem.



Morphological Analyzer Module

This module performs full morphological analysis of an input text.

Enter text here:

Бәрі жақсы болады.

Output:

```
бәрі_R_SIM |
жақсы_R_SE | жақсы_R_US | жақсы_R_ZE | жақсы_R_MOD |
бол_R_ET а_T1 ды_P3 | бол_R_ET а_T1 ды_P7 | бол_R_ETK а_T1
ды_P3 |
._R_NKT |
```

Fig. 1. A screenshot demonstrating a work of the morphological analyzer module.

Previously we have implemented a model for recognition of named entities in Kazakh language based on conditional random field (CRF) [28]. However, in our latest setup, we use a hybrid approach combining a bidirectional LSTM neural network and a CRF models. The main idea is to feed the features determined by CRF as input to LSTM network, thus, replacing the linear scoring by non-linear neural network scoring. The performance of this model in terms of F1-measure is 88%. For more details see our work [10,28].

4 Toolkit and Web Service

This project aims at building free/open source language processing tools for Kazakh. The proposed set of tools is designed to tackle a wide range of NLP problems that can be divided into pre-processing, core processing, and application-level routines. Each NLP task is implemented as a separate programming module in *Python 3* programming language and released under *CC-SA-BY* license. The current version of source code of KazNLP and the documentation are available on Github repository [15]. KazNLP can be installed and run on all platforms including Linux, OS X, and Windows, where Python is supported.

In addition to the packages, we have developed a web service that can be tested by an interested user. The web service is accessible online via link http://kazcorpus.kz/kaznlp/. Figure 1 shows a screenshot demonstrating a work of the morphological analyzer module.

5 Conclusion and Future Work

In this paper we presented a new package KazNLP which implements the core natural language processing tasks for automated processing of texts in Kazakh language. Rather than emphasizing the design and usage of the package, for which the reader is referred to the corresponding documentation in the repository, we focused on the NLP tasks and outlined the way we tackled them in order to better explain the internals of the developed modules.

Although KazNLP is a unique and holistic tool oriented for Kazakh language, we understand there is a considerable amount of work necessary to make it mature and widely used in research and industry. Therefore, as a future work, we plan to improve documentation, usability and performance of the modules as well as implement the latest methods and algorithms in the field of natural language processing.

Acknowledgments. Supported by Ministry of Education and Science of the Republic of Kazakhstan under the grants No. AP05134272 and No. AP08053085.

References

1. Assylbekov, Z., et al.: A free/open-source hybrid morphological disambiguation tool for kazakh (2016)
2. Baldwin, T., Lui, M.: Language identification: the long and the short of the matter. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 229–237. Association for Computational Linguistics, Los Angeles, California (2010)
3. Bird, S., Loper, E.: NLTK: the natural language toolkit. In: Proceedings of the ACL Interactive Poster and Demonstration Sessions, pp. 214–217. Association for Computational Linguistics (2004)
4. Evang, K., Basile, V., Chrupała, G., Bos, J.: Elephant: sequence labeling for word and sentence segmentation. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1426. Association for Computational Linguistics (2013)
5. Hakkani-Tür, D.Z., Oflazer, K., Tür, G.: Statistical morphological disambiguation for agglutinative languages. In: Proceedings of the 18th Conference on Computational Linguistics - Volume 1, pp. 285–291. Association for Computational Linguistics (2000)
6. Han, B., Baldwin, T.: Lexical normalisation of short text messages: makn sens a #twitter. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 368–378. Association for Computational Linguistics (2011)
7. Kessikbayeva, G., Cicekli, I.: Rule based morphological analyzer of kazakh language. In: Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM, pp. 46–54. Association for Computational Linguistics (2014)
8. Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.: OpenNMT: open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations, pp. 67–72. Association for Computational Linguistics, Vancouver, Canada July 2017 https://www.aclweb.org/anthology/P17-4012

9. Kozhirbayev, Z., Yessenbayev, Z., Makazhanov, A.: Document and word-level language identification for noisy user generated text. In: 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–4 (2018)
10. Kozhirbayev, Z., Yessenbayev, Z.: Named entity recognition for the kazakh language. Journal of Mathematics, Mechanics and Computer Science, 106(2) (2020) (Submitted)
11. Lui, M., Baldwin, T.: Langid.py: an off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations, pp. 25–30. Association for Computational Linguistics (2012)
12. Cheng, V., Li, C.H.: Combining supervised and semi-supervised classifier for personalized spam filtering. In: Zhou, Z.H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 449–456. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71701-0_45
13. Makazhanov, A., Sultangazina, A., Makhambetov, O., Yessenbayev, Z.: Syntactic annotation of kazakh: following the universal dependencies guidelines. a report. In: 3rd International Conference on Turkic Languages Processing, (TurkLang 2015), pp. 338–350 (2015)
14. Makazhanov, A., Yessenbayev, Z.: NLA-NU Kazakh Dependency Treebank. https://github.com/nlacslab/kazdet. Accessed 10 June 2020
15. Makazhanov, A., Yessenbayev, Z., Kozhirbayev, Z.: KazNLP: NLP Tools for Kazakh Language. https://github.com/nlacslab/kaznlp. Accessed 10 June 2020
16. Makhambetov, O., Makazhanov, A., Sabyrgaliyev, I., Yessenbayev, Z.: Data-driven morphological analysis and disambiguation for kazakh. In: Gelbukh, A. (ed.) CICLing 2015. LNCS, vol. 9041, pp. 151–163. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18111-0_12
17. Makhambetov, O., Makazhanov, A., Yessenbayev, Z., Matkarimov, B., Sabyrgaliyev, I., Sharafudinov, A.: Assembling the kazakh language corpus. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1022–1031. Association for Computational Linguistics (2013)
18. McDonald, R., Pereira, F.: Online learning of approximate dependency parsing algorithms. In: 11th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics (2006)
19. Müller, T., Cotterell, R., Fraser, A., Schütze, H.: Joint lemmatization and morphological tagging with lemming. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2268–2274. Association for Computational Linguistics (2015)
20. Mussayeva, A.: Kazakh language spelling with hunspell in openoffice.org. Tech. rep., The University of Nottingham (2008)
21. Myrzakhmetov, B., Yessenbayev, Z., Makazhanov, A.: Initial normalization of user generated content: case study in a multilingual setting. In: 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–4 (2018)
22. Myzakhmetov, B., Yessenbayev, Z.: Normalization of noisy user comments in kazakh language using statistical machine translation approach. Bulletin of the Eurasian National University (2020) (Submitted)
23. Nivre, J.: Universal Dependencies. https://universaldependencies.org/. Accessed 10 June 2020

24. Nivre, J., Hall, J., Nilsson, J.: MaltParser: a data-driven parser-generator for dependency parsing. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06). European Language Resources Association (ELRA) (2006)
25. Oflazer, K., Guzey, C.: Spelling correction in agglutinative languages. In: Fourth Conference on Applied Natural Language Processing, pp. 194–195. Association for Computational Linguistics (1994)
26. OpenNLP: The Apache OpenNLP Library. https://opennlp.apache.org/. Accessed 13 June 2020
27. Straka, M., Hajič, J., Straková, J.: UDPipe: trainable pipeline for processing CoNLL-u files performing tokenization, morphological analysis, POS tagging and parsing. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 4290–4297. European Language Resources Association (ELRA) (2016)
28. Tolegen, G., Toleu, A.: Named entity recognition for kazakh using conditional random fields. In: The 4-th International Conference on Computer Processing of Turkic Languages (TurkLang 2016) (2016)
29. Toleu, A., Tolegen, G., Makazhanov, A.: Character-aware neural morphological disambiguation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 666–671. Association for Computational Linguistics (2017)
30. Toleu, A., Tolegen, G., Makazhanov, A.: Character-based deep learning models for token and sentence segmentation. In: The 5-th International Conference on Computer Processing of Turkic Languages (TurkLang 2017) (2017)
31. Washington, J., Salimzyanov, I., Tyers, F.: Finite-state morphological transducers for three kypchak languages. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014), pp. 3378–3385. European Language Resources Association (ELRA) (2014)
32. Yessenbayev, Z., Kozhirbayev, Z.: Data-driven dependency parsing for kazakh. Bulletin of the Eurasian National University (2020) (Submitted)