УДК 81’32

CHARACTER-BASED DEEP LEARNING MODELS FOR TOKEN AND SENTENCE SEGMENTATION

Alymzhan Toleu[1,2], Gulmira Tolegen[1,2], Aibek Makazhanov[1],
[1]National Laboratory Astana, 53 Kabanbay Batyr ave., Astana, 010000, Kazakhstan
[2]Tsinghua University, Department of Computer Science and Technology, Beijing, 100084, China
aibek.makazhanov@nu.edu.kz

In this work we address the problems of sentence segmentation and tokenization. Informally the task of sentence segmentation involves splitting a given text into units that satisfy a certain definition (or a number of definitions) of a sentence. Similarly, tokenization has as its goal splitting a text into chunks that for a certain task constitute basic units of operation, e.g. words, digits, punctuation marks and other symbols for part of speech tagging. As seen from the definition, tokenization is an absolute prerequisite for virtually every natural language processing (NLP) task. Many of so called downstream NLP applications with higher level of sophistication, e.g. machine translation, additionally require sentence segmentation. Thus both of the problems that we address are the very basic steps in NLP and, as such, are widely regarded as solved problems. Indeed there is a large body of work devoted to these problems, and there is a number of popular, highly accurate off the shelf solutions for them. Nevertheless, the problems of sentence segmentation and tokenization persist, and in practice one often faces certain difficulties whenever confronted with raw text that needs to be tokenized and/or split into sentences. This happens because existing approaches, if they are unsupervised, rely heavily on hand-crafted rules and lexicons, or, if they are supervised, rely on extraction of hand-engineered features. Such systems are not easy to maintain and adapt to new domains and languages because for those one may need to revise the rules and feature definitions.

In order to address the aforementioned challenges, we develop character-based deep learning models which require neither rule nor feature engineering. The only resource required is a training set, where each character is labeled with an IOB (Inside Outside Beginning) tag. Such training sets are easily attainable from existing tokenized and sentence-segmented corpora, or, in absence of those, have to be created (but the same is true for rules, lexicons, and hand-crafted features). The IOB-like annotation allows us to solve both tokenization and sentence segmentation problems simultaneously casting them as a single sequence-labeling task, where each character has to be tagged with one of four tags: beginning of a sentence (S), beginning of a token (T), inside of a token (I)

and outside of a token (O). To this end we design three models based on artificial neural networks: (i) a fully connected feed forward network; (ii) long short term memory (LSTM) network; (iii) bi-directional version of LSTM. The proposed models utilize character embeddings, i.e. represent characters as vectors in a multidimensional continuous space.

We evaluate our approach on three typologically distant languages, namely English, Italian, and Kazakh. In terms of evaluation metrics we use standard precision, recall, and F-measure scores, as well as combined error rate for sentence and token boundary detection. We use two state of the art supervised systems as baselines, and show that our models consistently outperform both of them in terms of error rate.

Keywords: Token and Sentence Segmentation; Neural Networks; Deep Learning.

СИМВОЛЬНЫЕ МОДЕЛИ ГЛУБИННОГО ОБУЧЕНИЯ ДЛЯ ГРАФЕМАТИЧЕСКОГО АНАЛИЗА

Алымжан Толеу[1,2], Гульмира Толеген[1,2], Айбек Макажанов[1]
[1]Национальная Лаборатория Астана, пр. Кабанбай батыра 53, Астана, 010000, Казахстан
[2]Университет Цинхуа, Факультет Компьютерных Наук и Технологий, Пекин, 100084, КНР
aibek.makazhanov@nu.edu.kz

В настоящей работе мы рассматриваем задачу графематического анализа, а именно проблемы сегментации текста на предложения и токены. Сегментация текста по предложениям рассматривается как задача нахождения отрывков текста, удовлетворяющих одному или нескольким определениям предложения. Сегментация на токены (токенизация) – задача разбиения текста на операционные единицы, т.е. слова, цифры, знаки препинания и пр. Токенизация является базовой задачей обработки естественного языка (ОЕЯ). Большинство прикладных задач ОЕЯ, отличающихся относительной сложностью, например машинный перевод, нуждаются в сегментации входного текста по предложениям. Таким образом, обе рассматриваемые нами задачи являются основополагающими для ОЕЯ, и, как следствие, считаются в достаточной степени решенными. Действительно, опубликовано немало исследований по данной тематике, и существуют готовые решения широкого применения с хорошей точностью. Тем не менее, проблемы графематического анализа в большинстве случаев остаются открытыми, и на практике с ними приходится сталкиваться каждый раз, когда появляется необходимость в работе с необработанным текстом, т.е. не разбитым на предложения и токены. Это происходит потому, что существующие под-

ходы основаны либо на словарях и правилах (необучаемые), либо на извлечении вручную заданных признаков (обучаемые). Такие подходы тяжело адаптировать к новым языкам/жанрам, так как это требует переопределение словарей, правил и признаков.

Для снятия вышеупомянутых ограничений мы разработали символьные модели глубинного обучения, которые не нуждаются в определении правил или признаков. Единственное в чем есть необходимость – это обучающая выборка, в которой каждый символ помечен IOB меткой. Подобные обучающие выборки легко получить из имеющихся сегментированных и токенизированных корпусов. В случае отсутствия последних обучающую выборку придется создавать вручную, как в прочем, и словари и правила для других методов. Использование IOB разметки позволяет решать обе задачи одновременно, как одну задачу разметки последовательности, цель которой присвоить каждому символу одну из четырех меток: начало предложения (S), начало токена (T), тело токена (I), или пробел (O). Для решения данной задачи мы разработали три модели, основанные на искусственных нейронных сетях: (1) поступательная сеть; (2) LSTM сеть; (3) двунаправленная LSTM сеть. Разработанные модели используют символьные вложения, т.е. представления символов в виде векторов в многомерном пространстве.

Мы оцениваем наш подход на трех типологически отдаленных языках: английском, итальянском и казахском, используя стандартные метрики точности, покрытия, F-меры и процента ошибки. Для сравнения мы используем две широко распространённые системы графематического анализа, и показываем, что обе уступают нашим моделям по метрике процента ошибки.

Ключевые слова: Графематический анализ; нейронные сети; глубинное обучение.

1. Introduction

Let us begin by a quick recap of definitions. Sentence segmentation, aka sentence boundary detection, is a problem of segmenting a text into sentences for further processing; and tokenization is a problem of segmenting a text into chunks that for a certain task constitute basic units of operation (e.g. words, digits, etc.). At a first glance the problems seem trivial; after all, most written languages use special symbols to terminate sentences and whitespaces to delimit words. This is however not always the case.

First, although for many languages sentence final punctuation consists of a period (dot), a question and an exclamation mark, some languages use different sets of symbols (Brown, 2017). Second, regard-

less of symbols used as delimiters in any given language, chances are that those symbols have other functions as well, e.g. periods (dots) may be used in abbreviations, initials or in numbers as decimal points. Third, sentence and token definitions depend on the task at hand. For instance, while sentence segmentation may not be needed and a simple whitespace tokenization may be enough for a bag of word-based document classification, for parsing one may need to consider multiple sentence utterances in direct speech as a part of a host sentence (sentences in a sentence) and count clitics (syntactic words usually delimited with hyphens and apostrophes, but not whitespaces) as separate tokens. 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.

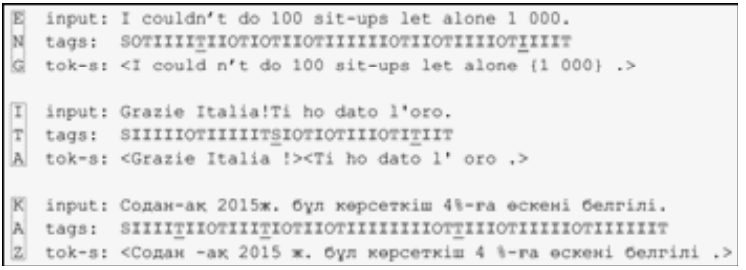

Fig. 1. An Example of an IOB-labeled text in English, Italian, and Kazakh

In this work we cast the token and sentence segmentation (TSS) problems as a single sequence labeling task and propose artificial neural network-based solutions, namely three character-based deep learning models. Unlike much of the previous work, our approach requires neither rule nor feature engineering. The only resource required is a training set, where each character is labeled with an IOB (Inside Outside Beginning) tag. Performing TSS jointly and using IOB-like format is not, in itself, a novelty, Evang et al. (2013) have implemented this approach in their CRF-based system called Elephant. However, unlike Elephant, our models make use of character embeddings, i.e. map characters into continuous vector space, and make no use of predefined features. Experiments show that our models achieve top performance for Kazakh language, for which TSS evaluation has never been carried out before. In order to show that the proposed models can achieve competitive results we compare them to a popular TSS system

Punkt (Kiss and Strunk, 2006) and the aforementioned Elephant system, which is considered the state-of-the-art. For this experiment we use publically available data sets for English and Italian languages.

2. Related Work

Existing systems for token and sentence boundary detection are based on hand written rules, unsupervised and supervised learning approaches. Rule-based systems (Grefenstette, 1999; Jurafsky and Martin, 2008; Dridan and Oepen, 2012) utilize hand-written rules, fixed lists of abbreviations and other lexical items to detect sentence boundaries. As a result such approaches are hard to maintain and not easy to adapt to new languages (Silla Jr. and Kaestner, 2004) or domains.

Unsupervised learning systems do not require specific hand-coded regular expressions and annotated training data. Mikheev (2002) presented an unsupervised approach for sentence boundary detection, proper name identification and abbreviation detection. The proposed system achieved respective error rates of 1.41% and 0.65% on WSJ and Brown corpora. The author concluded that the most crucial factor for sentence segmentation was detection of abbreviations and proper names. A similar system called Punkt was proposed by Kiss and Strunk (2006). The approach here has two detection stages: abbreviation detection and token-based classification. This system reached high accuracy, rivaling handcrafted rule-based and unsupervised systems. Compared with Mikheev's system, Punkt's error rates on WSJ and Brown corpora were 1.65% and 1.02%, respectively.

Supervised learning approaches utilize hand-engineered features, such as POS tags, tokens neighboring potential sentence boundaries, abbreviation lists, letter case (lowercase, uppercase), etc. These systems utilized maximum entropy models (Reynar and Ratnaparkhi, 1997) and conditional random fields (Fares et al., 2013). Many works have shown that conditional random field (CRF) is the most popular model for sequence labeling tasks (Lafferty et al., 2003; Tolegen et al., 2016).

Evang et al. (2013) presented a CRF-based TSS system – Elephant that uses a single character as a basic unit of operation. The system uses several features, such as Unicode categories, Unicode character codes, and combination of the two, as well as the 10 most active outputs of learned hidden states of a deep learning model as one feature category. Unlike our approach, Elephant uses the discrete features

rather than distributed embedding features. Numerous works on deep learning for NLP have shown the advantage of embeddings that tend to capture meaningful information and reduce task-specific features engineering.

3. Method

3.1 IOB labeling

In order to jointly learning one model for two tasks, we adopted IOB tagging scheme to identify the boundaries of the tokens and sentences. An example is given in Fig. 1. The tags S and T denote the beginnings of sentence and token boundaries respectively. Inside of a token is labeled I, and outside as O. Passages included in "<" and ">" denote segmented sentences. In the given example whenever tokens and sentence boundaries are not preceded by an outside character (O) they are underlined.

3.2 A general neural network

We introduce a general neural network model (Collobert et al., 2011) for token and sentence boundaries detection. The model is usually characterized by three specialized layers: (i) a character look-up table layer that extracts a window of character's embeddings from a character parameter matrix; (i) a general hidden layer; (iii) one output layer that is used to compute normalize scores for labels. The model architecture is shown on Fig. 2 and in what follows we refer to this model as NN.

Character look-up table. Let C be the list of characters derived from training data, d be the dimension of character embeddings, $Q \in R^{d \times |C|}$ be the matrix of character's embeddings. Suppose that a string s is made up of a sequence of characters $[c_1,\ldots,c_l]$, where l is the length of string. Then the character-level representation of s of is given by the matrix Q, where the j-th column of matrix Q corresponds to the character embedding for C_j. We use a sliding window approach to get a fixed sized w (a hyper-parameter) window of character embeddings around current character. Each character in the window is first passed through the look-up operation which produces a matrix of character embeddings that can be viewed as a w×d-dimensional vector x by concatenating each column vector, which can be fed into the next layers.

Hidden layer. The embedding of characters $x \in R^{w \times d}$ is extracted from the look-up table and is fed into a hidden layer which performs non-linear transformation followed by an element-wise activation function σ such as *tanh*, and the computation of this layer is:

$$h = \sigma(W_1 x + b_1) \quad (1)$$

where $W_1 \in R^{H_1 \times wd}$ is the parameter, $b_1 \in R^{H_1 \times 1}$ is a bias term, $h \in R^{H_1}$ is hidden units, H_1 is dimension of hidden layer.

The output layer is finally added on the top of the hidden layer for scoring boundary labels:

$$Score\ (x,T,\theta) = softmax(W_2\ h + b_2) \quad (2)$$

where $Score(x,y,\theta) \in R^{|T| \times 1}$ is a score of labels that computed by neural network with parameters $\theta = \{Q, W_1, b_1, W_2, b_2\}$, |T| is the number of tags. The parameters of models are initialized to small random numbers and automatically trained the by back-propagation algorithm.

3.3 Bi-directional LSTMs

Recently, LSTM neural networks have shown great promise in many NLP tasks (Greff et al., 2015; Ling et al., 2015; Toleu et al.,

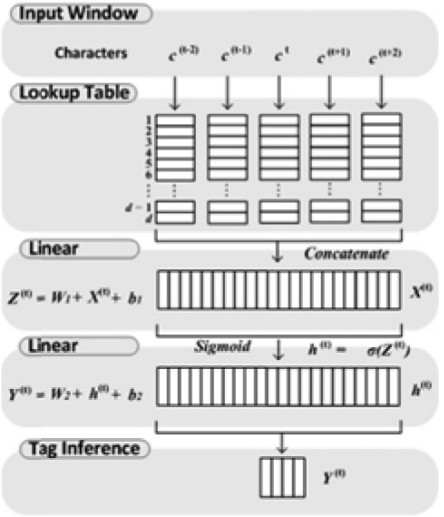

Fig. 2. Model architecture

2017) including language modelling, part-of-speech tagging etc. The architecture of LSTM consists of a set of recurrently connected states that can be viewed as memory blocks. Each block contains certain self-connected memory cells and three gates: input, output and forget gate. The gates provide continuous analogues of write, read and reset operation for the cells.

In order to examine the effectiveness of LSTM network for TSS, we use a model to predict each boundary label using LSTM. The architecture of our LSTM-based network is a variant that was described by Graves and Schmidhuber (2005), and is frequently cited in the literature.

Given a string made up of a sequence of characters , we encode each character into a vector representation then feed into our LSTM-based models, computing the forward hidden state and the backward hidden state. Both hidden states are concatenated into a single vector and fed into the output layer. In what follows we refer to this model as bi-LSTM, the model only uses the forward hidden states as LSTM. The architecture of the model is shown on Fig. 3.

4. Experiments

4.1. Data sets

The experiments were conducted on three datasets: (i) Kazakh texts from Kazakh corpus (Makhambetov et al., 2013) and UD treebank (Makazhanov et al., 2015); (ii) English newswire texts taken from the Groningen Meaning Bank, GMB (Basile et al., 2012); (iii) Italian texts from the PAIS`A corpus (Borghetti et al., 2011). Each dataset was split into three parts: a training set, a validation set, which is used for early stopping to select the best model and for optimizing the hyper-parameters, and a test set used for the final evaluation. Kazakh data needed additional processing as it was not IOB-labeled. We have performed an automatic IOB labeling based on existing token and sentence segmentations. Table 1 provides statistics on the domains of the texts and data quantities in terms of numbers of sentences and tokens.

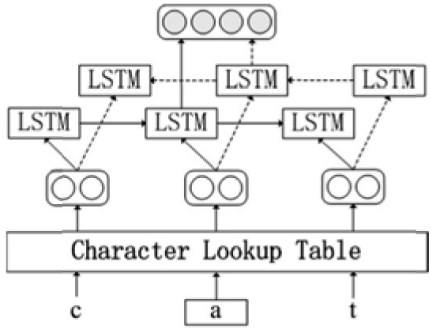

Fig. 3. Architecture of bi-LSTM-based model

Table 1. **Characteristics of the data sets**

Language	Domain	# sentences	# tokens
Kazakh	web/various	4 360	96,760
English	newswire	2 886	64,443
Italian	web/various	42 674	869,095

4.2. Model setup

We implement all neural network models using Java programming language and use the same hyper-parameters in all of three models: 35 for character level embeddings with random initialization, 9 for window size, 100 hidden states. We run 300 epochs on training and development sets, and select one model that is optimized on evaluation over the development set. The selected model is applied to the test set for the final evaluation. We used the CoNLL evaluation script to report, accuracy, precision, recall and F-measure over the token and sentence boundary labels.

4.3. Results

As it can be seen from Table 2, a general neural network model (NN) achieves a perfect 100 on all metrics, and clearly outperforms LSTM-based models in the task of sentence boundary detection for English language. One possible explanation is that the model NN has a window (the size is 9) to capture some corresponding characters and

predict a label to the centered one character, in this case, the prediction is made by conditioning on the left and right 4 characters. As our preliminary experiments showed that taking a smaller or larger window size, it harms the model performance on the sentence boundary label, but for token boundary, it does not have a significant effect.

Table 2. **Evaluation results for English**

	Sentence segmentation			Tokenization		
Models	Precision	Recall	F-measure	Precision	Recall	F-measure
NN	**100**	**100**	**100**	99.92	99.82	99.87
LSTM	99.34	99.34	99.34	99.94	**99.86**	**99.90**
bi-LSTM	99.67	99.34	99.50	**99.95**	**99.86**	**99.90**

On the other hand, the LSTM-based models achieve marginal improvement of the NN model in tokenization. In general all of the three models achieve near perfect results on the English data set.

Table 3. **Evaluation results for Italian**

	Sentence segmentation			Tokenization		
Models	Precision	Recall	F-measure	Precision	Recall	F-measure
NN	**99.28**	96.32	97.78	99.63	99.78	99.70
LSTM	99.00	96.27	97.62	99.52	99.71	99.61
bi-LSTM	99.25	**96.76**	**97.99**	**99.74**	**99.86**	**99.80**

As shown in Table 1, the size of the Italian data set is more than ten times larger than that of English and eight times larger than that of Kazakh. It is interesting to see the performance of neural network models for token and sentence boundary detections given larger training data. As evident from Table 3, the bi-LSTM model benefited the most from the abundance of data and was second to the NN model only in terms of precision of sentence segmentation. In general for the Italian language sentence segmentation turned out to be less accurate compared to English, but tokenization is still at the acceptable 99.8% in terms of F-measure.

From Table 4 one can observe that for Kazakh language the NN model detects sentence boundaries more accurately even though the other models use the same context window size (from the preliminary experiments, we observed that all of the LSTM-based models gave lower results without using a context window). This model has the highest recall in the tokenization task. As we have learned from the experiment on Italian, LSTM-based models are more sensitive to the size of training data, and thus maybe performing lower on a relatively small data set. In general all of the models exhibit a significantly lower performance on Kazakh data set. This can be explained by the fact that a large portion of this data set came from the Web (cf. Table 1), a notoriously noisy source. While the Italian data set contains certain amount of Web texts as well, this data set as a whole is much larger than the Kazakh one. Thus, we speculate that the fact that the data set was noisy and small may have hindered the performance of the models.

Table 4. **Evaluation results for Kazakh**

	Sentence segmentation			Tokenization		
Models	Precision	Recall	F-measure	Precision	Recall	F-measure
NN	**92.70**	**99.44**	**95.95**	99.74	**99.44**	99.59
LSTM	92.43	97.95	95.11	99.58	99.43	99.50
bi-LSTM	92.20	99.25	95.60	**99.82**	99.40	**99.61**

Table 5. **Comparison with other systems**

	English		Italian	
Models	Sentence (F-measure)	Sent. + Tok. (error rate)	Sentence (F-measure)	Sent. + Tok. (error rate)
Punkt	98.51	–	98.34	–
Elephant	**100**	0.27	**99.51**	0.76
NN	**100**	0.05	97.78	0.12
LSTM	**100**	**0.03**	97.62	0.13
bi-LSTM	**100**	**0.03**	97.99	**0.07**

In order to assess the performance of our models relative to existing systems we compare the performance of our models to the results reported by Evang et al. (2013) for their system Elephant and for another popular TSS system, Punkt (Kiss and Strunk, 2006). That is to say, we do not actually run and evaluate those systems. Instead we run our systems on the data that were used in the original experiment (Evang et al., 2013) and compare the results to the ones reported in that original experiment. The comparison is carried out in terms of F-measure of sentence boundary detection and combined (sentence and token segmentation) error rate. The results of the comparison are given Table 5.

As it can be seen, for English all of the models achieve a perfect F-measure of 100% on the sentence segmentation task, except Punkt that performs at 98.51%. When it comes to the combined TSS error rate our LSTM-based models achieve the lowest score of 0.03, improving 9 times over the state-of-the-art system, Elephant. When it comes to sentence boundary detection for Italian, however, our models are outperformed by both of the baseline systems. Here Elephant achieves a very strong F-measure of 99.51%, Punkt yields 98.34%, and the best of our models, bi-LSTM, performs at a decent 97.99%. Nevertheless, as it was the case with English, in terms of error rates for both token and sentence segmentation, our models perform much better, yielding the scores of 0.12, 0.13, 0.07 for NN, LSTM and bi-LSTM (without using any external features) respectively. Here the best performing model, bi-LSTM, improves almost 11 times over Elephant, whose error rate was 0.76. These results indicate that character-based deep learning models are better at modeling token boundary detection and also give very competitive results for sentence segmentation.

5. Conclusion

We have presented character-based deep learning models for joint token and sentence boundary detection. The main advantage of our approach is that it does not require any manual rule and feature engineering, and as such, is easy to maintain and adapt to new languages/domains. We have carried out both an absolute and comparative evaluation of our models on three languages (Kazakh, English and Italian). Our experiments showed that the proposed models achieve competitive results when compared to the state-of-the-art systems.

Acknowledgements

This work has been supported by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan under the targeted program O.0743 (0115PK02473), and by the Nazarbayev University under the research grant №129-2017/022-2017.

REFERENCES

1. Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In LREC 2012, pages 3196–3200, Istanbul, Turkey.

2. Claudia Borghetti, Sara Castagnoli, and Marco Brunello. 2011. I testi del web: una proposta di classificazione sulla base del corpus PAISA`. InM. Cerruti, E. Corino, and C. Onesti, editors, Formale e informale. La variazione diregistro nella comunicazione elettronica, pages 147–170. Carocci, Roma.

3. Brown T. Julian. Punctuation. https://www.britannica.com/topic/punctuation. Last accessed 09.09.2017.

4. Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, pages 2493–2537.

5. Rebecca Dridan and Stephan Oepen. 2012. Tokenization: Returning to a long solved problem – a survey, contrastive experiment, recommendations, and toolkit –. In ACL2012 (Volume 2: Short Papers), pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

6. Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence Labeling for Word and Sentence Segmentation. In EMNLP 2013, pages 1422–1426, Washington, USA.

7. Murhaf Fares, Stephan Oepen, and Zhang Yi. 2013. Machine learning for high-quality tokenization – replicating variable tokenization schemes. In CICLING 2013, pages 231–244.

8. Graves, A, Liwicki, M, Fernandez, S, Bertolami, R, Bunke, H, and Schmidhuber, J. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5), 2009.

9. Klaus Greff, Rupesh Kumar Srivastava, Jan Koutn´ık, Bas R. Steunebrink, and Jurgen Schmidhuber. ¨ 2015. LSTM: A Search Space Odyssey. CoRR abs/1503.04069.

10. Gregory Grefenstette. 1999. Tokenization. In Hans van Halteren, editor, Syntactic Wordclass Tagging, pages 117–133. Kluwer Academic Publishers, Dordrecht.

11. Daniel Jurafsky and James H. Martin. 2008. Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, 2nd edition.

12. Kiss, Tibor, and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. Computational Linguistics 32.4, pages 485–525

13. John D. Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In ICML ’01, pages 282–289.

14. Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Lus Marujo, and Tiago Lus. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In EMNLP2015, pages 1520–1530.

15. Makazhanov A., Sultangazina A., Makhambetov O., Yessenbayev Z. 2015. Syntactic Annotation of Kazakh: Following the Universal Dependencies Guidelines. A report. TurkLang 2015, pages 338–350.

16. Makhambetov O., Makazhanov A., Yessenbayev Z., Matkarimov B., Sabyrgaliyev I., Sharafudinov A. 2013. Assembling the Kazakh Language Corpus. In EMNLP2013, pages 1022–1031.

17. Andrei Mikheev. 2002. Periods, capitalized words, etc. Computational Linguistics, 28(3):289–318.

18. Carlos N. Silla Jr. and Celso A. A. Kaestner. 2004. An analysis of sentence boundary detection systems for English and Portuguese documents. In CICLing 204, pages 135–141.

19. Toleu, Alymzhan and Tolegen, Gulmira and Makazhanov, Aibek. 2017. Character-Aware Neural Morphological Disambiguation. In Proceedings of ACL 2017, pages 666–671,Vancouver, Canada.

20. Gulmira Tolegen, Alymzhan Toleu, and Zheng Xiaoqing. 2016. Named Entity Recognition for Kazakh Using Conditional Random Fields. In Proceedings of TurkLang 2016, pages 122–129.

21. Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In Proceedings of ANLP97, pages 16–19.