Postprint: Kazakh Syntactic Analysis Based on Sentence Spans

Authors: Chai Wei, Gulila · Adongbieke

Date: 2019-01-03T00:00:00+00:00

Abstract

Due to the currently low accuracy of Kazakh syntactic parsing and the lack of relevant research on neural network-based Kazakh syntactic analysis, this study addresses phrase-structure syntactic parsing for Kazakh using a shift-reduce approach. However, it stores sentence spans in the stack rather than partial tree structures, thereby eliminating the need for syntactic tree binarization during parsing. For sentence feature extraction, this study employs bidirectional LSTM to extract sentence span features, obtaining contextual information of sentence spans within the entire sentence. A multilayer perceptron is then used to train the syntactic parsing model, and dynamic programming is utilized during decoding to select the optimal parsing result. This approach ultimately achieves an accuracy of 76.92% for Kazakh phrase syntactic parsing. The research findings contribute to further improvements in Kazakh syntactic parsing accuracy and lay a solid foundation for subsequent Kazakh machine translation and semantic analysis.

Full Text

Preamble

Research on Kazakh Syntactic Parsing Based on Sentence Spans

Chai Wei[1],[2],[3], Gulila Altenbek[1],[2],[3]
(1. College of Information Science & Engineering, Xinjiang University, Urumqi 830046, China;
2. Xinjiang Laboratory of Multi-language Information Technology, Urumqi 830046, China;
3. The Base of Kazakh & Kirghiz Language of National Language Resource Monitoring & Research Center on Minority Language, Urumqi 830046, China)

Abstract: Due to the low accuracy of Kazakh syntactic parsing and the lack of research on neural network-based approaches, this paper addresses phrase-

structure parsing for Kazakh using a shift-reduce method that stores sentence spans rather than partial tree structures in the stack, thereby eliminating the need for binarization during parse tree construction. The model employs bidirectional LSTM to extract sentence span features, capturing contextual information across the entire sentence, and uses a multilayer perceptron to train the parsing model. Dynamic programming is applied during decoding to select the optimal parsing result, achieving a final phrase-structure parsing accuracy of 76.92% for Kazakh. These results represent a significant improvement in Kazakh syntactic parsing accuracy and provide a solid foundation for subsequent Kazakh machine translation and semantic analysis.

Keywords: Bi-LSTM; sentence span; dynamic oracle

0 Introduction

Syntactic parsing automatically derives the grammatical structure of a sentence according to a given grammatical system, analyzing the grammatical units contained in the sentence and their relationships to transform the sentence into a structured syntax tree [1]. As a crucial bridging component in natural language processing, syntactic parsing provides a essential foundation for deeper tasks such as machine translation and semantic role labeling.

In early rule-based parsing research, linguists manually crafted grammatical rules. For example, Collins and Hallet et al. [2,3] employed context-free grammar rules or scored sentences based on generated rules to produce phrase-structure trees. In statistical parsing research, Charniak [4] proposed a pure PCFG-based parsing method and subsequently developed a lexicalized PCFG approach. Currently, neural network methods have substantially improved parsing accuracy, reducing the importance of explicit grammatical rules. For instance, Dyer and Stern [5,6] utilized encoder-decoder models where the encoder reads and represents sentences as vectors, and the decoder generates labeled syntax trees from these vectors.

Previous research has explored Kazakh phrase-structure parsing. Reference [7] implemented a self-learning Kazakh phrase-structure parser using PCFG with Viterbi algorithm in the decoding module. Reference [8] combined probabilistic context-free grammar with the Chart algorithm to develop a PChart-based Kazakh phrase-structure parser. Reference [9] employed PCFG with perceptron-based reranking to implement a coarse-to-fine Kazakh phrase-structure parser.

Although Kazakh syntactic parsing has been studied using rule-based and statistical methods, the advancement of neural network technology has yielded increasingly accurate models for sequential prediction. This paper applies such techniques to Kazakh parsing by modifying the shift-reduce system to store sentence spans instead of partial tree structures in the stack, using bidirectional LSTM to capture contextual span features. Bidirectional LSTM obtains

sentence-level information, capturing the context of sentence spans within the entire sentence. Additionally, dynamic programming is integrated into Kazakh syntactic parsing, enabling optimal parsing results without reranking during decoding. The advantage of sentence spans over partial tree structures lies in their representation of sub trees corresponding to sentence fragments, where the head word must be located at either the beginning or end of the span, thereby reducing the search space complexity during decoding.

The main contributions of this paper are: (a) applying neural networks to Kazakh syntactic parsing with favorable results; (b) utilizing sentence spans as the basic unit stored in the stack for shift-reduce parsing; (c) employing bidirectional LSTM to capture sentence context information; and (d) using dynamic programming to obtain optimal parsing results without reranking during decoding. In this system, reduction is always prioritized when both shift and reduce actions are possible.

1 Kazakh Syntactic Parsing System

Kazakh belongs to the Turkic branch of the Altaic language family and is an agglutinative language. Based on its characteristics, Kazakh currently employs a two-level phrase annotation system with five basic phrase types and ten part-of-speech tags at the first level [8].

Recent work [10,11] has proposed a novel parsing approach that relies on extracting information directly from surface sentence spans rather than through rich grammatical structures. A phrase-structure syntax tree can be viewed as a collection of labeled spans within a sentence. Guided by this principle, we propose a model consisting of two components: one that scores label assignments to sentence spans, and another that directly scores the existence of sentence spans. The former determines output labels, while the latter provides the output span structure. The core challenge in both components is the representation of sentence spans. Since the correct labeling of a span depends heavily on its context, we leverage neural networks, which have proven effective at capturing contextual information suitable for various natural languages.

The parsing process employs a shift-reduce method as the fundamental model, involving two primary data structures: an input queue and an analysis stack. Here, σ represents the analysis stack storing sentence spans corresponding to phrases being analyzed bottom-up; z records the number of shift-reduce parsing steps; i and j denote elements to be analyzed in the stack; X represents a unary chain or non-terminal node. Since there is only one labeling action choice, parsing a sentence of n words requires (4n-2) steps; t represents the collection of sentence spans, which contains the complete syntax tree upon finishing parsing.

The parsing process involves four main actions:

a) **Shift**: Push <word, POS> from the queue onto the stack.

b) **Combine**: Merge the top two sentence spans on the stack.

c) **Label-X**: Label the top span of the stack.

d) **Nolabel**: Perform no labeling.

The first two are structural actions, while the latter two are phrase-labeling actions. Phrase-labeling actions primarily convert one or more non-terminal sentence spans in the stack into partial tree structures.

This approach resembles the odd-even parsing method proposed in [12], but since the stack stores sentence spans rather than partial tree structures, no binarization is required during training or parsing. For example, given a sentence, Figure 1 [Figure 1: see original paper] shows its gold parse tree. Table 1 illustrates the static decoding process: steps 1-2 perform shift actions with no labeling since they do not form a phrase; steps 5-6 combine the top two spans into a phrase and label it (label-VP), with other steps following similarly. This static decoding method generates "short-stack" style syntax trees, where both shift and reduce actions can produce results.

The parsing input consists of word and POS tag vectors, with POS tags provided as input to the parser. Word vectors can be initialized randomly or pre-trained on large corpora; this work uses random initialization. The neural network architecture is shown in Figure 2 [Figure 2: see original paper]: after extracting span features using a 2-layer bidirectional LSTM, a multilayer perceptron is employed for training. Each word is represented by a d-dimensional vector, and POS tags are similarly represented by d-dimensional vectors. The input passes through ReLU activation functions to map words and POS tags to hidden layers, where bias terms are included. A softmax layer atop the hidden layer calculates multi-class probabilities to determine the most likely label, with |L| representing the number of phrase labels.

2 Sentence Span Feature Extraction

Reference [13] demonstrated that a single-layer bidirectional LSTM can represent word positions in sentence context, while a 2-layer bidirectional LSTM further improves parsing performance by feeding the two-layer positional feature representations into a fully connected layer. This work employs a 2-layer bidirectional LSTM to obtain contextual information for an input i encoded in forward and backward directions. The representation of sentence span (i, j) is computed through vector differences combined with the shift-reduce system for parsing. As shown in Figure 3 [Figure 3: see original paper], sentence spans are modeled by computing vector differences. In this model, the sentence is input once, and the same recurrent outputs are used throughout the parsing process to compute span features.

When considering structural actions during parsing, the decision of whether to combine the top two spans is crucial; for labeling actions, the top stack span

is an important candidate. Therefore, four span features determine structural actions, while three span features determine phrase-labeling actions, enabling precise sentence segmentation. In both cases, prefix and suffix of the remaining input are also considered. The feature templates are shown in Table 2 .

3 Dynamic Programming

Dynamic programming has been used and proven optimal for span labeling in [14-16]. It makes labeling decisions based on two principles: (a) if the label assigned to a sentence span is part of the gold syntax tree, it is correct; (b) otherwise, the label is empty.

For structural decisions, dynamic programming operates on syntax tree branch points. If sentence span (i, j) appears in the gold tree, b(i, j) represents its internal boundary points. For example, span (1,8) with sub-spans (1,3), (3,5), and (5,8) has internal boundary points b(1,8) = {3,5}. Dynamic programming precisely outputs spans consistent with the gold tree. If span (i, j) does not correspond to the gold tree, it identifies the minimal similar span, where minimality depends on a partial order induced by span length. The dynamic programming output is the set of internal boundary points enclosing the span, which also lies within the original internal boundary points.

4 Experiments

4.1 Experimental Data and Evaluation Metrics

The experimental data is derived from the *Xinjiang Daily* (Kazakh edition), with all corpora manually annotated and proofread. The data is split in an 8:1:1 ratio for training, development, and testing. The experimental results are evaluated using the PARSEVAL metric, primarily computing labeled precision (LP), labeled recall (LR), and F-measure (F_1) to assess parsing quality. The evaluation functions are defined as:

$$\mathrm{LP} = \frac{\text{Number of correctly parsed phrases}}{\text{Total number of parsed phrases}} \times 100\%$$

$$\mathrm{LR} = \frac{\text{Number of correctly parsed phrases}}{\text{Total number of phrases in gold treebank}} \times 100\%$$

$$\mathrm{F_1} = \frac{2 \times \mathrm{LP} \times \mathrm{LR}}{\mathrm{LP} + \mathrm{LR}} \times 100\%$$

4.2 Experimental Environment

The experiments are implemented in Python using the LSTM neural network model from the DyNet toolkit [17]. DyNet, developed by Google and Carnegie Mellon University among others, is a dynamic neural network toolkit for natural language processing that includes common models such as RNN, CNN, and

LSTM. The hardware environment consists of 32 GB RAM, an NVIDIA GeForce GTX 1080Ti GPU, and a 64-bit Ubuntu 16.04 operating system.

4.3 Experimental Results and Analysis

The proposed Kazakh parsing model is influenced primarily by word vector features and sentence length. The following experiments investigate these factors.

4.3.1 Impact of LSTM Feature Extraction on Parsing Accuracy When using Bi-LSTM for sentence feature extraction, the number of hidden layer nodes affects feature quality. To examine this impact, experiments were conducted with hidden layer sizes of 50, 70, 100, 120, 150, and 200. The results are shown in Figure 4 [Figure 4: see original paper].

The results demonstrate that: (a) hidden layer size affects feature extraction and consequently parsing performance; (b) beyond 120 hidden units, increasing size yields diminishing returns. Therefore, subsequent experiments use a hidden layer size of 120.

4.3.2 Impact of Sentence Length on Parsing Accuracy Sentence length significantly affects parsing performance. The test corpus was divided by sentence length into categories of 1-5 words, 6-15 words, and over 15 words. The results are presented in Table 3 .

The results show that shorter sentences achieve better parsing accuracy than longer ones, primarily because shorter sentences contain fewer nested phrases and simpler syntax tree structures.

4.3.3 Parser Performance Comparison To evaluate the proposed model, its performance was compared against existing Kazakh parsers. The neural network structure parameters are listed in Table 4 , and training hyperparameters in Table 5 .

Table 4 Neural Network Structure Parameters - Word vector dimension: 100 - POS vector dimension: 50 - LSTM layers: 2 - LSTM hidden dimension (per direction): 120 - ReLU hidden units (per action): 200

Table 5 Training Hyperparameters - Vector initialization: Random initialization - Gradient descent rate: = 0.01 - Adaptive learning rate: = 0.0001 - Dropout: p = 0.5

Using the parameters from Tables 4 and 5, the trained parser was tested and compared with results from [7-9], as shown in Table 6 .

Table 6 Comparison of Performance of Different Parsers on Kazakh Test Set

Method	LP (%)	LR (%)	F_1 (%)
Method from [7]	68.36	67.89	68.12
Method from [8]	70.29	69.87	70.08
Method from [9]	71.42	71.03	71.22
Our method	76.92	76.53	76.72

The comparison demonstrates that span-based Kazakh phrase-structure parsing with dynamic programming significantly improves accuracy, achieving gains of 8.56%, 6.63%, and 5.5% over PCFG, PChart, and PCFG-perceptron methods, respectively. The use of bidirectional LSTM for span feature extraction proves highly effective for Kazakh phrase-structure parsing.

5 Conclusion

This paper investigates Kazakh phrase-structure parsing based on sentence spans. Compared to previous methods using fixed feature templates for contextual information, bidirectional LSTM effectively captures contextual span features, improving parsing accuracy. The integration of dynamic programming into Kazakh syntactic parsing yields excellent results without requiring reranking during decoding. Future work will focus on increasing training corpus size and exploring different neural network architectures to further enhance Kazakh phrase-structure parsing accuracy.

References

[1] Allen J F. Natural Language Understanding [M]//Natural Language Understanding. [S.l.]: Benjamin Cummings Pub. Co., 1995: 617-626.
[2] Colins M. Three generative, lexicalised models for statistical parsing [C]//Proc of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics. Stroudsburg, PA: Association for Computational Linguistics, 1997: 16-23.
[3] Hall D, Durrett G, Dan K. Less Grammar, More Features [C]//Proc of Meeting of the Association for Computational Linguistics. Stroudsburg, PA: Association for Computational Linguistics, 2014: 228-237.
[4] Charniak E. Statistical parsing with a context-free grammar and word statistics [C]//Proc of the 14th National Conference on Artificial Intelligence and the 9th Conference on Innovative Applications of Artificial Intelligence. AAAI Press, 1997: 598-603.
[5] Dyer C, Kuncoro A, Ballesteros M, et al. Recurrent neural network grammars[C]//Proc of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 199-209.
[6] Stern M, Andreas J, Dan K. A minimal span-based neural constituency parser [C]//Proc of Meeting of the Association for Computational Linguistics.

Stroudsburg, PA: Association for Computational Linguistics, 2017: 818-827.
[7] Shang Wenqing, Gulila · Altenbek, Niu Na et al. Syntactic Analysis of Kazakh Language Based on PCFG Model [J]. Modern Computer, 2015(5): 7-10.
[8] Shang Wenqing, Gulila Altenbek, Niu Na et al. Kazakh syntactic analysis based on probabilistic Chart algorithm [J]. Computer Engineering and Design, 2016, 37(3): 832-836.
[9] Liang Jinlian, Gulila Altenbek. A coarse-to-fine Kazakh PSG parse [J]. Journal of Chinese Information Processing, 2018, 32(1): 83-88.
[10] Hall D, Durrett G, Dan K. Less grammar, more features [C]//Proc of Meeting of the Association for Computational Linguistics. Stroudsburg, PA: Association for Computational Linguistics, 2003: 228-237.
[11] Cross J, Huang L. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles[C]//Proc of Conference on Empirical Methods in Natural Language Processing. 2016: 1-11.
[12] Chen D, Manning C D. A fast and accurate dependency parser using neural networks [C]//Proc of Conference on Empirical Methods in Natural Language Processing. 2014: 740-750.
[13] Cross J, Huang Liang. Incremental parsing with minimal features using bi-directional LSTM [EB/OL]. (2016-08-07). https://www.aclweb.org/anthology/P/P16/P16-2006.pdf.
[14] Mi Haitao, Huang Liang. Shift-reduce constituency parsing with dynamic programming and POS tag lattice [C]//Proc of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015: 1030-1035.
[15] Goldberg Y, Nivre J. Training deterministic parsers with non-deterministic oracles [EB/OL]. (2013-10-12). http://aclweb.org/anthology/Q/Q13/Q13-1033.pdf.
[16] Ballesteros M, Goldberg Y, Dyer C, et al. Training with exploration improves greedy stack-LSTM parser (2016-10-27). http://www.aclweb.org/anthology/D/D16/D16-1211.pdf.
[17] Neubig G, Dyer C, Goldberg Y, et al. DyNet: the dynamic neural network toolkit [EB/OL]. (2017-01-15). https://arxiv.org/pdf/1701.03980.pdf.

Note: Figure translations are in progress. See original paper for figures.