doi:10.3772/j.issn.1006-6748.2019.01.011

Gated recurrent unit model for a sequence tagging problem[①]

Rekia Kadari[②] *, Zhang Yu, Zhang Weinan, Liu Ting
(Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, Harbin 150001, P. R. China)

Abstract

Combinatory categorial grammer (CCG) supertagging is an important subtask that takes place before full parsing and can benefit many natural language processing (NLP) tasks like question answering and machine translation. CCG supertagging can be regarded as a sequence labeling problem that remains a challenging problem where each word is assigned to a CCG lexical category and the number of the probably associated CCG supertags to each word is large. To address this, recently recurrent neural networks (RNNs), as extremely powerful sequential models, have been proposed for CCG supertagging and achieved good performances. In this paper, a variant of recurrent networks is proposed whose design makes it much easier to train and memorize information for long range dependencies based on gated recurrent units (GRUs), which have been recently introduced on some but not all tasks. Results of the experiments revealed the effectiveness of the proposed method on the CCGBank datasets and show that the model has comparable accuracy with the previously proposed models for CCG supertagging.

Key words: combinatory categorial grammer (CCG), CCG supertagging, deep learning, gated recurrent unit (GRU)

0 Introduction

Combinatory ctegorial grammar (CCG) forms an important class of lexicalized grammar formalism. In CCG grammar, each word is associated with a lexical syntactic category. The syntactic CCG categories have two types: atomic (Sentence (S), Noun (N), Noun Phrase (NP) and Prepositional Phrase (PP)), and complex categories that are built by combining atomic categories or complex categories themselves using slashes; ('\') or ('/') indicating whether the argument should appear on the left or on the right, respectively.

Sequence labeling corresponds to several tasks of natural language processing (NLP) problems that aim at associating a sequence of symbols (i. e. words) with labels (syntactic and semantic information) facilitating many NP-complete problems such as parsing, question answering, among others. CCG supertagging is an important task in NLP that can be treated as a sequence labeling problem and formulated as follows: given a sentence of N words $\{W_1, W_2, \cdots, W_n\}$; the supertagger aims at assigning a lexical category y to each word in the sentence[1] $\{y_1, y_2, \cdots, y_n\}$. CCG supertagging is a fundamental step that should be performed before full parsing and remains a significant challenge due to the huge number of the associated lexical categories to each word.

High performance approaches have been dominated to this challenging problem. Reducing the number of the assigned lexical categories was the main focus of several researchers that have proposed different methods based on machine learning approaches such as the maximum entropy models[2] and the feed forward neural network by Lewis and Steedman[3]. However, these methods have many drawbacks and require many features.

In recent years, deep learning methods have achieved massive success in many NLP tasks such as machine translation [4], language modeling[5], sequence-to-sequence learning tasks[6], handwriting recognition and generation[7], and various tagging problems[8]. As an alternative to the feed forward neural networks and to overcome the drawbacks of the model proposed by Lewis and Steedman [3], Xu et al. [9] have demonstrated the effectiveness of the recurrent neural networks (RNNs) for the CCG supertagging problem

① Supported by the National Basic Research Program (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61472105, 61502120).

② To whom correspondence should be addressed. E-mail: rekia@ir.hit.edu.cn

Received on Jan. 25, 2018

and have achieved promising results over several state-of-the-art models.

However, the difficulty of training RNNs to capture long-term dependencies[10] has encouraged various promising research attempts to solve it. Different variants of RNNs have been proposed. In this paper, a variant of recurrent nets based on gated recurrent units (GRUs)[11] is proposed which is able to retain information for long dependencies, to solve the problem of CCG supertagging. As the task can be treated as a sequence prediction problem where we can access to information in both previous and future directions, the proposed model is centered around bidirectional recurrent networks and more specifically is a bidirectional gated recurrent unit (Bi-GRU) model that has the ability to memorize information not only for one direction but also in both past and next directions of a word eliminating the problem of limited context[7] and also for long periods of time.

Experiments on CCGBank datasets were conducted. The results are compared with most of the state-of-the-art models. The results on CCG supertagging task show that the GRU model outperforms most existing models such as maximum entropy models[2], feed forward neural networks[3], and RNNs[9] and demonstrate that bidirectional RNNs are of a vital importance for CCG supertagging as a sequence tagging task. The best results remain that the Bi-GRU based model has reached an accuracy of 93.87% on the test data.

Rest of the paper is organized as follows: in Section 1 related works are discussed. Next, Section 2 introduces GRU architecture used in this paper for CCG supertagging. In addition, Section 3 gives details about the experiments and provides experimental results along a comparison with previous works. Finally, in Section 4 the paper is concluded.

1 Related work

CCG supertagging is an essential technique to perform as a primary step for many NLP tasks. For example, in the parsing task, the supertagging is a crucial step because it deals with reducing the search space that the parser must explore. Recently, CCG supertagging has become a very important task in the NLP area and attracted the attention of several researchers in the NLP community. In literature, dominant approaches that exploited machine learning and deep learning models have been developed. In this section we briefly introduce some proposed models for the CCG supertagging task.

Clark and Curran[2] used a set of lexical categories that appeared at least 10 times in Sections 02-21 of the CCGBank resulting in 425 categories and introduced maximum entropy (ME) models using words and part of speech (POS) tags in the five-word window, plus the two previously assigned lexical categories to the left as features. Because the number of the assigned lexical categories could be high, Clark and Curran[2] used a tag dictionary: the supertagger assigned categories with the word in the data for words seen at least k times and assignsed categories with the POS tag in the data to the words seen less than k times. Unfortunately, this model has a number of shortcomings. First, its high reliance on POS tags leads to low accuracy on out-of-the-domain datasets[12]. Second, because the main set of features is based on raw words and POS tags, there is degradation in the performance of the model in the presence of rare and unknown words[12]. Third, in order to reduce the feature sparsity, every tagging decision is made without considering any contextual information over a local context window.

Lewis and Steedman[3] proposed a feed forward neural network model to deal with the drawbacks of Clark and Curran's[2] model mentioned above that heavily relied on features by using word embeddings that was more accurate to predict lexical categories without dependence on POS tags with no lexical or POS features. The features used in Lewis and Steedman's[3] model were similar to those used by Collobert et al.[13] for POS tagging based on character suffixes and whether a word is capitalized or not in a context window performing some data preprocessing (e. g. all words are lower-cased).

Recently, deep learning emerges as an effective way to solve many NLP problems and overcome several shortcomings of machine learning methods and have achieved good results on various tasks. Xu et al.[9] exploited RNNs for CCG supertagging that have the ability to capture information for long distance outperforming state-of-the-art methods and addressing all previously mentioned problems.

Following Lewis and Steedman[3], Xu et al.[9] used the same set of features with a window size of seven and achieved good and comparable results to the state-of- the-art methods.

However, traditional RNNs suffer from several limitations[10] where gradients tend to either vanish or explode in earlier layers over long sequences. A variant of sophisticated recurrent units proposed more recently by Cho et al.[11] is the gated recurrent unit (GRU). In this paper, the gated recurrent unit architecture is introduced to solve the CCG supertagging problem.

The model uses both previous and future information between input words via bidirectional GRU achieving comparable results to the state-of-the-art. The results verify the effectiveness of GRU for CCG supertagging as a sequence labeling problem with a deep bidirectional architecture that is more convenient to learn complex interactions between entries for both previous and next directions of an input sequence.

2 Proposed method

In this section, RNNs are described briefly before the description of the proposed model.

Recurrent neural networks are a class of artificial neural networks that have the ability to make use of sequential information performing the same task for every element in a sequence, where the output at each time step is dependent on that at the previous time step. RNNs have been successfully used for many applications such as language modeling[5], spoken language understanding[14] and CCG supertagging[9].

The structure of the widely used RNNs models is introduced by Elman[15] which consists of a hidden layer h_t that is updated based on input x_t and the previous hidden state h_{t-1} providing an output layer y_t.

In CCG supertagging, x_t represents input features and y_t represents the predicted lexical categories. The RNN computes hidden layer h_t and output layer y_t as follows:

$$h_t = f(U\,x_t + W\,h_{t-1}) \tag{1}$$

$$y_t = g(V\,h_t) \tag{2}$$

where U, W and V are the connection weights and g is the activation function.

In theory, RNNs are designed to store history information for long sequences; however, in practice it is hard to train traditional RNNs that capture information for long periods and sequences because of the vanishing/exploding problems. To deal with these problems many variants of RNNs have been proposed, among which the gated recurrent units represent a good alternative.

2.1 Gated recurrent units

Recently, as an alternative to the simple RNNs, the gated recurrent unit architecture was proposed by Cho et al.[11] to solve the difficulty of training traditional RNNs. The main idea of GRUs is that it has a memory cell which decides the degree of information to keep in the memory from the previous states and it is known to be good at preserving long distance dependencies. GRU networks have gates to control the information:

(1) a reset gate r: determines how to combine the new input with the previous memory and decide whether the past sequence is relevant for the future or not;

(2) an update gate z: defines how much of the previous memory information to keep around. Fig. 1 illustrates all these components.

Mathematically, a GRU hidden state h_t given an input x is calculated as is described by the equations below:

$$z_t = \sigma(W_z[h_{t-1},\ x_t] + b_z) \tag{3}$$

$$r_t = \sigma(W_r[h_{t-1},\ x_t + b_r) \tag{4}$$

$$\tilde{h}_t = tanh(W_h[r_t \odot h_{t-1},\ x_t] + b_h) \tag{5}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{6}$$

where σ is the logistic sigmoid function, r and z are the reset and update gates, respectively, $\odot$ stands for element-wise multiplication, W is the weight matrices and the b terms denote the bias vectors.

Fig. 1 illustrates a gated recurrent unit where r and z are the reset and update gates, respectively, and h and $\tilde{h}$ are the activation and candidate activation.

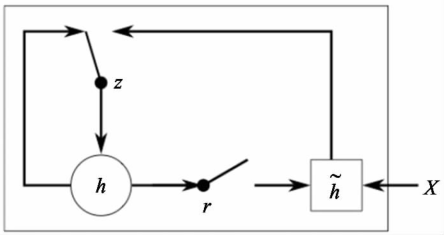

Fig. 1 A gated recurrent unit memory block

2.2 GRU proposed model for CCG supertagging

In sequence labeling tasks, information is accessed on both past and future directions of an input sequence. Therefore, it is reasonable to use models that are able to capture previous and future input information. An elegant solution to model sequential data that has achieved high accuracies in many sequence labeling tasks is bidirectional models. The idea behind bidirectional models is to present each sequence in two separate layers to capture past and future information respectively, which is well-suited for our task. The two outputs from each sequence are then concatenated to form the final output.

This paper is interested in using GRU networks for CCG supertagging. The proposed model consists of a bidirectional GRU where a forward GRU computes the hidden sequence ($\overrightarrow{h_t}$) and reads an input from the beginning to the end and a backward GRU ($\overleftarrow{h_t}$) uses the opposite direction. The outputs from each GRU (backward and forward) are then fed to another backward and forward GRU layers. Finally the outputs from each layer at each time step are concatenated [$\overrightarrow{h_t}$, $\overleftarrow{h_t}$] and fed through a Softmax layer to decode them into probabilities for each supertag forming the final output of the network.

A deep architecture that is more convenient in capturing complex interactions in the context between words is used. The architecture of the model is shown in Fig. 2.

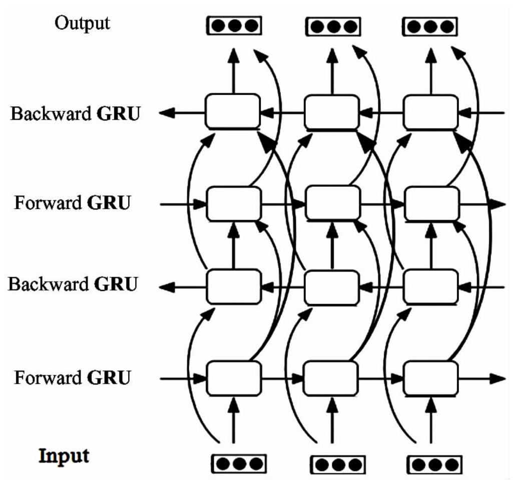

Fig. 2 Model overview

The network uses look-up tables of feature vectors that are first concatenated and then fed to the network. The output at each time step is fed through a Softmax layer to decode it into probabilities for each Supertag.

Word-level features: the final model uses four sets of features.

Following Lewis and Steedman[3], suffixes and capitalization features are used.

Capitalization feature: The capitalization feature has only two values indicating whether a given word is capitalized or not.

Suffix feature: State-of-the-art existing CCG supertaggers are followed by using suffixes of size two.

Word embeddings: Given a sentence of N words $\{W_1, W_2, \cdots, W_n\}$, the embedding feature of w_t, W: words$\rightarrow R_n$ is a paramaterized function mapping words in some language to high dimensional vectors and is obtained by projecting it into an n-dimensional vector space using the look-up table. Each dimension describes syntactic or semantic properties of the word. It has shown that word embeddings have been exceptionally successful and played a vital role in improving many NLP tasks performance such as sequence tagging[13]. For the experiments, word embeddings are used to improve the performance in solving the CCG supertagging problem.

- **Word index embeddings**: Task specific word embeddings model is used because several misspellings words, abbreviations, and compositions of words have occurred in the training data. These words are identified as 'UNKNOWN' words by a pre-trained word embeddings model. The task-specific word embedding model is built using the 'Embedding' layer of Keras[16] library [17]. The embedding layer takes a 2-dimensional matrix of integers representing each word in the corpus (index of the word in the corpus) as input and outputs a 3-dimensional matrix, which represents the word embedding model that maps the integer inputs to the vectors found at the corresponding index in the embedding matrix[16].
- **Pre-trained word embeddings**: The best model uses pre-trained Google's Word2Vec 300-dimensional embeddings which was trained on 100 billion words from Google news [18].

Following Collobert et al. [13], all words are lowercased before passing through the look-up tables to convert them into their corresponding embeddings and also all numbers are replaced by a single digit '0'. For a word that does not have an entry in the pre-trained word embeddings, the 'UNKNOWN' entry from the pre-trained embeddings is used.

3 Experiments and results

In this section, the data sets and training parameters of the proposed experiments are reported. The achieved results are then discussed where experiments are conducted to evaluate the model by applying it to supertagging and multi-tagging.

3.1 Experimental data

Following the standard split, the models are trained on Sections 2-21 of the CCGBank corpus[19] using section 00 (1 913 sentences) for development. The experiments tested the utility of the proposed models on section 23 from CCGBank corpus (2 407 sentences) as a testing set, and all the datasets are preprocessed as follows:

- All words are lowercased.
- All sequences of digits are transformed into a single digit '0'.
- For words and numbers containing '\/', back-off to the sub-string after the delimiter.

3.2 Hyper-parameters and training

The neural network is implemented using the version 1.2.2 of keras[16]; a Theano based neural network library. Training and testing are done on the sentence level. For pre-trained word-embeddings, the Google's Word2Vec[20] 300-dimensional embeddings are used. During development, experiments are done with Turian 100-dimensional embeddings but no remarkable improvements were seen on the resulting model. The accuracy of the model is tested on the development set with the hidden dimension values ranging in {100, 200, 256, 300, 400, 512, 600} and it is found that the hidden dimension with size 300 can provide the best accuracy.

3.2.1 Learning algorithm

The training was done by the SGD optimizer with a fixed learning rate of 0.01. Other more sophisticated optimization algorithms have been explored such as Adam and AdeDelta[21] without any remarkable improvement over SGD. For the output layers, the Softmax activation function is used.

3.2.2 Dropout

Dropout of fixed rate of 0.2 is applied to the input layer[22] that was quite effective in reducing over-fitting and gave significant improvements on accuracy.

3.3 Results and analysis

In this section, the evaluation of the performance of the proposed Bi-GRU model is presented for CCG supertagging on the CCGBank datasets, and multi-tagging experiments are also performed and the results are discussed bellow.

3.3.1 Supertagging results

The models for 90 epochs are trained and the parameters that gave the highest accuracy on the development set are used. The hyper-parameters are tuned then the models are trained. The final chosen parameters are reported in Table 1.

Table 1 The final chosen hyper-parameters for the best model

Hyper-parameter	Value
Word embeddings	Google's word2Vec
Hidden dimension	300
Dropout	0.2
Optimizer	SGD
Learning rate	0.01

The model is compared with the three systems previously proposed for CCG supertagging: the C&C model proposed by Clark and Curran[2] with gold and auto POS tags, the feed forward neural network (NN) by Lewis and Steedman[3], and the RNN network model by Xu et al.[9]. Table 2 compares the results with those models on the section 00 from the CCGBank corpus (development set).

By examining Table 2, it becomes obvious that the proposed Bi-GRU achieves higher accuracy over RNN model with an improvement of +0.40% and +0.9% over C&C with gold POS. Therefore, the use of GRU can lead to a better performance than simple recurrent networks and that the use of Bi-GRU was very useful to model and memorize more information from both directions of an input entry.

Table 2 1-best tagging accuracy on CCGBank Section 00 (development set)

Model	Accuracy (%)
C&C(gold pos)[2]	92.60
C&C(auto pos)[2]	91.50
NN[3]	91.10
RNN[9]	93.07
GRU	93.47

The overall results on the test set are shown in Table 3. Bi-GRU model improves the performance of CCG supertagging task to a significant extent, bringing up the accuracy from 91.57% to 93.87% comparing to feed forward NN by Lewis and Steedman[3].

Table 3 1-best accuracy on the test set

Model	Sec-23 (%)
C&C(gold pos)[2]	93.32
C&C(auto pos)[2]	92.02
NN[3]	91.57
RNN[9]	93.00
GRU	93.87

Also, it outperformed the RNN model proposed by Xu et al.[9] with a significant improvement. This may be due to the higher quality of the network that can learn from past and future entries, which helps the model to make more accurate predictions.

3.3.2 Multi-tagging results

The proposed model is also evaluated for multi-tagging where the proposed supertagger is able to assign more than one category to each word whose probabilities are within a β factor.

The performance of the proposed model on multi-tagging is measured in terms of WORD accuracy where the word is considered to be tagged correctly if the correct category is included in the set of the assigned lexical categories and SENT (sentence) accuracy which is the percentage of sentences whose words are all tagged correctly using the default β levels used by the C&C parser[2] on the development set. The results of these experiments are presented in Table 4. It can be seen that the model results have much better performance than the previous models for both WORD and SENT accuracies on all β levels.

Table 4 Performance comparison of different models for multi-tagging accuracy on Section 00 for different β levels. The Word column gives the WORD accuracy (%) and the SENT column gives the sentence accuracy (%)

β	GRU		RNN		NN		C&C (auto pos)		C&C (gold pos)	
	WORD	SENT	WORD	SENT	WORD	SENT	WORD	SENT	WORD	SENT
0.075	97.22	67.22	97.33	66.07	96.83	61.27	96.34	60.27	97.34	67.43
0.030	98.08	74.90	98.12	74.39	97.81	70.83	97.05	65.50	97.92	72.87
0.010	98.71	81.87	98.71	81.70	98.54	79.25	97.63	70.52	98.37	77.73
0.005	99.01	85.04	99.01	84.79	98.84	83.38	97.86	72.24	98.52	79.25
0.001	99.42	90.92	99.41	90.54	99.29	89.07	98.25	80.24	99.17	87.19

Backward GRU is very powerful in capturing past information on long time memorizing previous context information. On the other hand, forward GRU is also very efficient on memorizing future information on long periods. However, it is well known that single direction GRU suffers weakness of not utilizing the contextual information from the other direction of an input. Bidirectional GRUs utilize both the previous and future context by processing the sequence on two directions, one processes the input sequence in the forward direction, while the other processes the input from the future direction and generates two GRU output vectors, which was suitable for the task. This work demonstrates that bidirectional GRU architectures are capable of modeling sequential data and reaches high accuracy over simple recurrent models. The improvements are due to the deep bidirectional GRU architecture advantages capturing the interaction between sequences in two directions.

4 Conclusion

Gated recurrent unit (GRU) model is introduced (GRU) for CCG supertagging. The proposed approach uses look-up tables of features. Experiments are performed on the CCGBank datasets with the accuracy as an evaluation metric. Experiment results show that the proposed approach achieved state-of-the-art performances.

It is also found that: (1) traditional recurrent neural networks are extremely weak in modeling sequential data, while adding neural gates dramatically boosts the performance; (2) bidirectional GRU performs better than simple ones to capture information from two directions; (3) deep architecture is more convenient to capture interactions between inputs.

It would be beneficial to integrate the supertagger to the C&C parser where the supertagger is likely to play a significant role which will be left as a future work.

References

[1] Bangalore S, Joshi A K. Supertagging: An approach to almost parsing[J]. *Computational linguistics*, 1999, 25(2):237-265

[2] Clark S, Curran R J. Wide coverage efficient statistical parsing with CCG and log-linear models[J]. *Computational linguistics*, 2007, 33(4):493-552

[3] Lewis M, Steedman M. Improved CCG parsing with semi-supervised supertagging[J]. *Transactions of the Association for Computational Linguistics*, 2014, 2: 327-338

[4] Cho K, Merriënboer B V, Bahdanau B, et al. On the properties of neural machine translation: Encoder-decoder approaches[J]. 2014a, arXiv preprint arXiv:1409.1259

[5] Mikolov T, Anoop D, Stefan K, et al. RNNLM-recurrent neural network language modeling toolkit[C]. In: Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU) Waikoloa, 2011. 11-15

[6] Sutskever I, Vinyals O, Le Q. Sequence to sequence learning with neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, Montreal, Canada, 2014. 3104-3112

[7] Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks[C]. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, 2013. 6645-6649

[8] Dyer C, Ballesteros M, Ling W, et al. Transition-based dependency parsing with stack long short-term memory [J]. 2015, arXiv preprint arXiv:1505.08075

[9] Xu W, Auli M, Clark S. CCG supertagging with a recurrent neural network[C]. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL), Beijing, China, 2015. 250-255

[10] Bengio Y, Simard P, Frasconi P. Learning long term dependencies with gradient descent is difficult[J]. *IEEE Transactions on Neural Networks*, 1994, 5(2):157-166

[11] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation[J]. 2014b, arXiv preprint arXiv:1406.1078

[12] Rimell L, Clark S. Adapting a lexicalized grammar parser to contrasting domains[C]. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-08), Waikiki, USA, 2008. 475-484

[13] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. *The Journal of Machine Learning Research*, 2011, 12:2493-2537

[14] Mesnil G, Dauphin Y, Yao K, et al. Using recurrent neural networks for slot filling in spoken language understanding[J]. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2015,23(3):530-539

[15] Elman J. Finding structure in time[J]. *Cognitive science*, 1990, 14(2):179-211

[16] Chollet F. Keras. https://github. com/fchollet/keras: Github, 2015

[17] Lazib L, Zhao Y, Qin B, et al. Negation scope detection with recurrent neural networks models in review texts [C]. In: Proceedings of the International Conference of Young Computer Scientists, Engineers and Educators, Singapore, 2016

[18] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]. In: Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, USA, 2013. 3111-3119

[19] Hockenmaier J, Steedman M. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank[J]. *Computational Linguistics*, 2007, 33(3):355-396

[20] Turian J, Ratinov L, Bengio Y. Word representations: a simple and general method for semi-supervised learning [C]. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 2010. 384-394

[21] Zeiler M D. ADADELTA: an adaptive learning rate method[J]. 2012, arXiv preprint arXiv: 1212.5701

[22] Pham V, Bluche T, Kermorvant C, et al. Dropout improves recurrent neural networks for handwriting recognition. In: Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), Crete, Greece, 2014. 285-290

Rekia Kadari, is a Ph. D candidate at the Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China. She received her B. S. and M. S. degrees in computer science from Dr. Tahar Moulay University of Saida, Algeria in 2011 and 2013 respectively. Her research interest include natural language processing, sequence labeling, and CCG supertagging.