Available online at www.ijpe-online.com

International Journal of Performability Engineering

Vol. 13, No. 5, September 2017, pp. 775-782
DOI: 10.23940/ijpe.17.05.p20.775782

An Attention-Based Syntax-Tree and Tree-LSTM Model for Sentence Summarization

Wenfeng Liu[a,b], Peiyu Liu[a,c,*], Yuzhen Yang[b], Yaling Gao[d], Jing Yi[a,e]

[a]*School of Information Science and Engineering, Shandong Normal University, Jinan, 250014, China*
[b]*Department of Computer and Information Engineering, Heze University, Heze, 274015, China*
[c]*Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan,250014, China*
[d]*Ruipu Peony Industrial Technology Development Co., Ltd, 274015, China*
[e]*School of Computer Science and Technology, Shandong Jianzhu University, Jinan, 250101, China*

Abstract

Generative Summarization is of great importance in understanding large-scale textual data. In this work, we propose an attention-based Tree-LSTM model for sentence summarization, which utilizes an attention-based syntactic structure as auxiliary information. Thereinto, block-alignment is used to align the input and output syntax blocks, while inter-alignment is used for alignment of words within that of block pairs. To some extent, block-alignment can prevent structural deviations on the long sentences and inter-alignment is capable of increasing the flexibility of the generation in the blocks. This model can be easily trained to end-to-end mode and deal with any length of the input sentences. Compared with several relatively strong baselines, our model has achieved the state-of-art on DUC-2004 shared task.

Keywords: Syntax-Tree; Tree-LSTM; Text Summarization

(Submitted on January 29, 2017; Revised on April 12, 2017; Accepted on June 23, 2017)

1. Introduction

Dealing with summarization, currently, there are two main methods, one is the extractive and the other is the generative. Summarization is a very challenging task in natural language understanding. In addition, most of the successful summary systems utilize the extractive way to extract the core content from the text for a simplified representation. Syntaxnet [1], Google's latest syntax parser [16, 21], provides a superduper assist for the generation of sentence-level summarization, which is very similar with the generation of traditional header line. Cohn and Lapata [6] propose a compressed method that allows more transformation operations, while Woodsend et al. [17] present a quasi-synchronous grammar method which is based on free text parsing and deterministic parsing to generate summary. Our work has been inspired by a neural network-based attention model [14] and neural headline generation with minimum risk training [2].

Furthermore, Wubben et al. [18] directly take advantage of MOSES for text simplification. Filippova and Altun [7] focus on the extractive compression and present a method to automatically build a compression corpus with hundreds of thousands of instances. Bahdanau et al. [3] propose an attention-based model. Their approach needs to search for parts of a source sentence that are relevant to predicting a target word. Utilizing LSTM (long short term memory) units and attention mechanism, Lopyrev [10] trains an encoder-decoder recurrent neural network for generating news headlines.

We focus on sentence-level summarization. Many models for this task are based on deleted or compressed techniques [8]. There is still a big gap in contrast with humans who generally use rewrite, generalization, or reordering to achieve that. Combining Word Embeddings, we propose an attention-based Tree-LSTM method for sentence-level summarization. In particular, we exploit attention-based syntax tree to obtain the syntactic alignment of the input and output pairs. Between the two pairs, block-alignment is used for the syntax blocks, while inter-alignment is employed inside the blocks. Block-alignment

* Corresponding author. Tel.: +86-183-6417-0677.
E-mail address: liupycn@163.com.

can prevent structural deviations on the long sentences, and the inter-alignment can increase the flexibility of the generation in the blocks.

2. Method

Figure 1 (a) is a blocked example of our approach about the sentence "Chinese President Xi met with Russian President Vladimir Putin at the Great Hall of the People." As shown in Figure 1 (b), the sentence is blocked on the basis of syntactic structure (the red arrows indicate the order in the sentence). Ultimately, we can get the above summarization.

According to the syntax structure, each input or output sentence is divided into a number of syntax blocks. With that, the block-alignment between blocks and inter-alignment inside the blocks are learned during the training. In the test, we select the K-highest block summaries representing one input block. The beam search is performed on all the blocks to obtain the optimal result as summarization of the sentence (as shown in Figure 1 (b)). The sentence "Chinese President Xi met with Russian President Vladimir Putin at the Great Hall of the People" is converted into "Xi meets with Putin in Beijing". The word "Beijing" has a high similarity with that of "the Great Hall of the People", so the long noun phrase "the Great Hall of the People" is replaced by a word "Beijing".

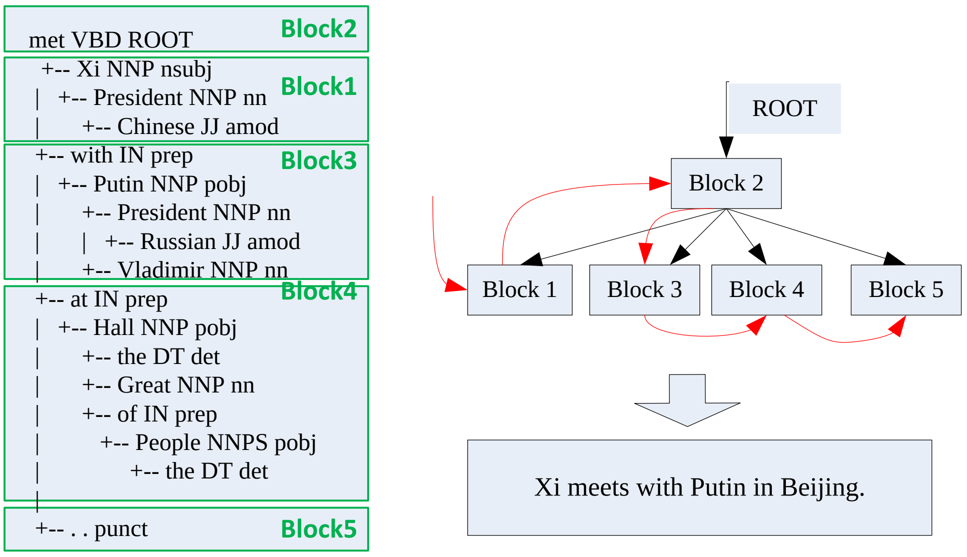

Figure 1. (a) An example of our approach using syntax-tree; (b) An example of Generative Summarization

In order to test the effectiveness of the method, we compare our model with many abstractive and extractive models, including information retrieval, machine translation based on phrase statistics, attention-based model combining with neural networks, etc. Our model has achieved the state of art on DUC-2004 shared task.

Given an input sentence, our goal is to get a simplified summarization. The input sentence $X = x_1,..., x_m$ comes from a fixed vocabulary *V*. The output is $Y = y_1,...,y_n$, Obviously, n<m. In our experiment, the summarization has the same vocabulary *V* as the input. Then we consider the issue of generating a summary, the log conditional probability can be formulized as:

$$\log P(y\,|\,x;\theta) = \sum_{b=1}^{B}\left(\sum_{k=1}^{N}\log P(y_k\,|\,x_b, y_{1...k-1};\theta)\right) \tag{1}$$

According to the syntax structure (as shown in Figure 1 (b)), the input sentence is divided into a number of blocks *B*. Similarly, the output has the same number of blocks as the input (Some output blocks may be empty). The *k*-th word y_k in summary is generated according to $y_{1...k-1}$ and block x_b. θ is the set of parameters. Each output block has multiple possible results. In order to avoid the problem of combination explosion, we select *K-highest* output summarization for each input block (using Eq.2).

$$k\_Max(x_{block}, y_{block}) \tag{2}$$

Here x_{block} is a syntax block of input sentence, y_{block} is the simplification of x_{block}. During the training, the input and output syntax blocks are obtained through section 3.1.

3. Model

The network diagram for our model is shown in Figure 2. During the training, the input and output sentence pairs are fed into the Syntax Tree model. Then we utilize attention-based model (section 3.3) to align the generated syntax tree pairs. Finally, we put these pairs together into the tree-LSTM to get the parameters of our model. In the testing, only the input sentences are needed, the outputs are summaries of these sentences.

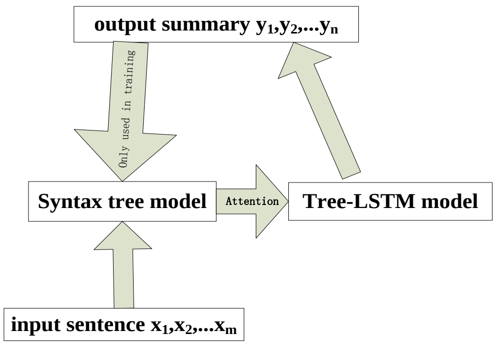
output summary $y_1,y_2,\ldots y_n$
Only used in training
Syntax tree model
Attention
Tree-LSTM model
input sentence $x_1,x_2,\ldots x_m$

Figure 2. The network diagram for our model

3.1 Syntax-Tree model

Syntax-Tree model, which is composed of a *STACK*, a *BUFFER* and a set of *ARCS*, is a transition-based dependency parser [4, 13, 20] that is incrementally constru-cted. It handles words from left to right. The unprocessed words of the sentence are put in *BUFFER* and part-of-speech (POS) are put in a list. The processed words are pushed into the *STACK*, and the *ARCS* describes the dependencies between words.

Table 1. Three operations of the parser

operation	description	example
op_shift	pushes the next_word*(w_2)* of the current_word(w_1) onto the top of the *Stack*.	*Stack:...w_1 Buffer: w_2,w_3,... pos=[w_1.pos, ...,w_n.pos]* *Run one op_shift operation* *Stack: ... w_1 w_2 Buffer: w_3,...*
op_left_arc	pops two words(w_1 w_2) on the top from the *Stack*, attaches the second(w_1) to the first*(w_2)*, creates an arc pointing to the left, pushes the first word back onto the *Stack*.	*Stack: ... w_1 w_2 Buffer: w_3,... pos=[w_1.pos, ...,w_n.pos]* *Run one op_left_arc operation* *Stack: Buffer: Arcs:* *... ... w_2 ↙ w_3, ... {..., (w_2,left_arc (w_1))}* *w_1*
op_right_arc	pops the top two words(w_1 w_2) from the *Stack*, attaches the second(w_1) to the first*(w_2)*, creates creates a right arc, and pushes the second(w_1) back onto the *Stack*.	*Stack: ...w_1 w_2 Buffer: w_3,... pos=[w_1.pos, ...,w_n.pos]* *Run one op_left_arc operation* *Stack: Buffer: Arcs:* *... ... w_1 ↘ w_3, ... {..., (w_1,right_arc (w_2))}* *w_2*

At the beginning, all the words of a sentence are placed in *the BUFFER*. At each step, the parser can only do one of the three operations:*op_shift*, *op_left_arc* and *op_right_arc(as shown in* Table 1*)*. Using this strategy, we can obtain the sentence syntax trees.The following is our Syntax-Tree model.

```
**Algorithm** Syntax_Tree_model ()
{//The initial configuration of Syntax_Tree model:
STACK stack = [root]
BUFFER buffer_words = [$w_1$,...,$w_n$]
POS pos=[$w_1$.pos, ..., $w_n$.pos] // POS={NN,JJ,VBZ,…}
ARCS arc_set ={ empty} //ARCS={dobj,amod,nsubj,…}
while(the buffer is not empty and the stack contains more than one node)
{ // The following operations are performed according to the words and POS of the top stack and the buffer
            If(the shift will be used in the transition) then { run op_shift operation }
      If(the left arc will be used in the transition)
            then { run op_left_arc operation;
            add a left arc about the two words to the arc_set.}
      If(the right arc will be used in the transition)
            then { run op_right_arc operation;
                  add a right arc about the two words to the arc_set.}
 }
}//the analysis ends.
```

Given an input sentence [w_1,...,w_n], we define a set of states *S* (s* denotes the start state). However, the most important thing is to define the decision function *D(s).* According to the state information(word and POS on top of the stack or in the buffer), *D(s)* will deside to run one of the three operations(*op_shift,op_left_arc,op_rig-ht_arc*).

In order to improve the efficiency of implementation, we use two stacks: one for handling words and one to deal with their part of speech. In addition, two buffers are needed. Two words on the top of the stack and two in the buffer as well as their part of speech are considered when deciding which operation will be executed. According to that operation, the two stacks and two corresponding buffers are performed in the same processing. We use the following feature combination templates, {stack_word1,stack_word2,buffer_word1,buffer_word2, stack_pos1,stack_pos2,buffer_pos1,buffer_pos2}.

Each combination of features corresponds to one of the three operations.

Finally we get the dependency tree of the words and the dependency tree of the part of speech about a sentence. According to the obtained syntax trees, different syntax blocks have different weights. In the syntax block, the closer the block center word is, the higher the weight. It will be detailed in section 3.3. And the weighted word embedding is fed into N-ary Tree-LSTM(section 3.2).

3.2 N-ary Tree-LSTM

N-ary Tree-LSTM[15] is a tree structure: each node has up to N branches, the child is ordered, and each node is indexed in order to store the results generated using syntax-tree model(3.1). For any node *j*, its *kth* child's hidden state and storage state are h_{jk} and c_{jk} respectively. The transformation of Tree-LSTM is as follows: for simplicity, the following formulas eliminate bias.

$$h_{jl} = \left[\overrightarrow{h_{jl}}; \overleftarrow{h_{jl}}\right]^T \tag{4}$$

$$\mathrm{i}_j = \sigma\left(\mathrm{W}^{(i)} x_j + \sum_{l=1}^{N} U_l^{(i)} h_{jl}\right) \tag{5}$$

$$\mathrm{f}_{jk} = \sigma\left(\mathrm{W}^{(f)} x_j + \sum_{l=1}^{N} U_{kl}^{(f)} h_{jl}\right) \tag{6}$$

$$\mathrm{o}_j = \sigma\left(\mathrm{W}^{(o)} x_j + \sum_{l=1}^{N} U_l^{(o)} h_{jl}\right) \tag{7}$$

$$\mathrm{u}_j = \tanh\left(\mathrm{W}^{(u)} x_j + \sum_{l=1}^{N} U_l^{(u)} h_{jl}\right) \tag{8}$$

$$\mathrm{c}_j = \mathrm{i}_j \odot \mathrm{u}_j + \sum_{l=1}^{N} \mathrm{f}_{jl} \odot c_{jl} \tag{9}$$

$$\mathrm{h}_j = \mathrm{o}_j \odot \tanh(\mathrm{c}_j) \tag{10}$$

Where x_j represents one word embedding of input sentence, σ denotes logistic sigmoid function, $\odot$ represents element-by-element multiplication. Each tree-LSTM cell at step *j* has an input gate i_j, a forget gate f_j, an output gate o_j , a memory cell c_j, and a hidden state h_j. *W* and *U* are parameter matrices.The forget gate controls the extent of forgetting about the previous memory cell, the input gate regulates how much each unit will be updated, and the output gate controls the exposure of the internal memory state. Therefore, the hidden state vector in a Tree-LSTM unit is a gated, partial view of the internal memory cell. For example, we consider a tree node whose left child corresponds to a block of noun phrase, and right child represents a block of verb phrase. Suppose that we will emphasize the verb phrase, the $U_{kl}^{(f)}$parameters can be trained that the components of $\mathrm{f}_{j1}$are very close to 0 (negligible), and the components of $\mathrm{f}_{j2}$are very close to 1 (means very important). Note that when the tree is simplified to a chain, Eqs.5-10 will be reduced to the standard LSTM transitions. In this way, we can obtain which word or block is more important in generating a summary.

3.3 Attention-based Syntax tree model

Using the attention-based Syntax tree model, we can obtain the syntactic alignment for the input and output sentences. Between the two pairs, block-alignment is used to align the syntax blocks, while inter-alignment is used to align inside the block pairs. Through this way, we can train input and output sentence pairs. Additionally, block-alignment and inter-alignment are very important parts of our model. Block-alignment can prevent structural deviations on long sentences, and the inter-alignment can increase the flexibility of the generation in the inner-blocks. As is shown in Figure 3, the hollow double arrows are the block-alignments between input and output syntax blocks. The alignment probability of input and output is directly obtained from their syntax tree. The red double arrows are the inter-alignments in the block pair, α_{io} is attention parameter matrix.

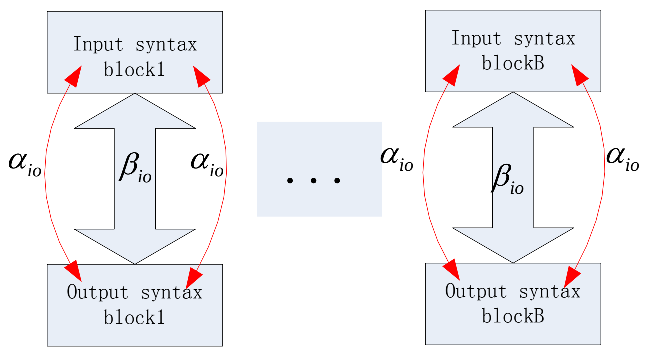

Figure 3. Attention mechanism of the input and output blocks

First of all, about the input block and its corresponding output block, we need to compute the cosine similarity of theirs (see Eq.(11)).

$$\beta_{io} = SIM(\vec{e}_i, \vec{e}_o) = \frac{\vec{e}_i \bullet \vec{e}_o}{\| \vec{e}_i \| \| \vec{e}_o \|} \tag{11}$$

Where $\vec{e}_i$ denotes the embeddings of one input block; $\vec{e}_o$ represents embeddings of a corresponding output block.

$$\vec{e}_i = \sum_{p \in input\_block} \lambda_p \vec{e}_{ip} \tag{12}$$

Where $\vec{e}_{ip}$ is a word embedding of input_block, $\sum_{p \in input\_block} \lambda_p = 1$, λ is a weight value.

For output block:

$$\vec{e}_o = \sum_{q \in output\_block} \omega_q \vec{e}_{oq} \tag{13}$$

Where $\vec{e}_{oq}$ is a word embedding of a output block, $\sum_{q \in output\_block} \omega_q = 1$, ω is a weight value. Wether λ or ω, their value depends on the location in the syntax tree, the closer to the root, the more important. The next section we describe the details of training and generating summary.

4. Training and Generating Summary

In this section, we describe details of generation. According to the syntactic composition of sentence and utilizing section 3.1, we block the input and output (output only blocked in training). Attention-based syntax trees are used for initial parameters. Then we feed the initial parameters and word embeddings to n-ary Tree-LSTM. The parameters are updated during training. However, during testing, each input block may correspond to multiple output blocks. In order to avoid the combinatorial explosion, we choose *k_Max* to represent output block.

The semantic of a block is a weighted average by the word embeddings in the block. Each word has a weight, the weight value is computed by exponential decay according to the distance from the word to the center word in the syntactic structure. Since each syntax block has k results, we use the beam search to get the best result.

$$x^1_{block\_i}, \ldots, x^1_{block\_i} = k\_Max(SIM(\vec{e}_i, \vec{e})) \tag{14}$$

Where $\vec{e}_i$ represents a syntactic block of input sentence, $\vec{e}$ is a vector of the word embeddings space (*WEspace*) . The attention-based syntactic structure model and word embeddings are used to simplify the input(see section 3.3). The scoring function is based on the semantic similarity of y_{block_j}.

In addition, there are no constraints for generation, so it can be trained in any pairs of the input and output. A training set consisting of pairs is defined as $\mathrm{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(J)}, y^{(J)})\}$.

Minimize the negative log-likelihood estimator parameters for the summary by using the following formula:

$$NLL(\theta) = -\frac{1}{J}\sum_{j=1}^{J} \log p(y_{block}^{1}, y_{block}^{2} \ldots y_{block}^{q} \mid x_{block}^{1}, x_{block}^{2} \ldots x_{block}^{q}; \theta) \tag{15}$$

Where q denotes the number of the sentence block. Minimum Risk Training (MRT) strategy is used in Eq.16.

$$M = \sum_{(x,y)\in D} \sum_{y' \in (x,\theta)} NLL(\theta)\Delta(\mathrm{y}', \mathrm{y}) \tag{16}$$

$\Delta(\mathrm{y}', \mathrm{y})$ denotes the KL-distance between y and generated summary y'. In order to simplify and improve the efficiency of training, we draw a random subset of *D* for MRT. For the purpose of reducing the number of parameters, the number of chunks q is consistent across all sentences. So some blocks may be *NULL*.

5. Experiments

5.1 Data Set and Word Embedding

DUC-2004 shared task is a standard evaluation dataset of sentence summarization. The data for this task consists of 500 news articles from the New York Times (NYT) and Associated Press Wire Services (APWS) each paired with 4 different human-generated reference summaries. The full data set is available by request at *http://duc.nist.gov/data.html*.

We also report evaluation on single reference headline-generation using a randomly subset of Gigaword, which is pre-processed with Stanford CoreNLP tools [3]. Gigaword contains around 9.5 million news articles sourced from various domestic and international news services over the last two decades. Word Embeddings are trained from Wiki2014 (about 1.6B token) by *Gensim*.

5.2 Baselines

On account of the diversity of the summary generative approaches, we use multiple headlines to generate the baselines.

Information Retrieval (IR) [12], the baseline indexes the training set and uses the highest BM-25 match as the title of the article.
COMPRESS [5] uses a language model to generate a compressed output on the header data using the syntactic structure of the source sentence.
TOPIARY [19] incorporates linguistic-driven transformations and an unsupervised topic detection algorithm (UTD).
MOSES+ [9] is based on a Statistical Machine Translation Model.
ABS and **ABS+** [6]**,** ABS is an attention Model for Sentence Summarization. ABS+ is a tuned model about ABS.

Table 2. Experimental results with ROUGE

		DUC-2004			**Gigaword**	
Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
IR[4,18]	11.06	1.67	9.67	16.91	5.55	15.58
COMPRESS[4,19]	19.77	4.02	17.30	19.63	5.13	18.28
TOPIARY[4,20]	25.12	6.46	20.12	-	-	-
MOSES+[4,21]	26.50	8.13	22.85	28.77	12.10	26.44
ABS[4]	26.55	7.06	22.05	30.88	12.22	27.77
ABS+[4]	28.18	8.49	23.81	31.00	12.65	28.34
This work	**28.46**	**9.02**	**24.07**	**31.42**	**12.98**	**28.92**

5.3 Implementation and Results

The most widely used evaluation metric for document summarization is ROUGE. The basic idea of ROUGE is to count the number of overlapping units between auto-generated summary and the reference summary, such as overlapped word sequence, n-grams and word pairs. It is the most common evaluation metric in Document Understanding Conference (DUC). What's more, we use a mini-batch stochastic gradient descent to minimize negative log likelihood. We use the *learning rate= 0.05*; *Word Embedding dimension =200*; *Hidden layer neurons= 200*; *momentum=0.9*; *batch_size=64*. If the validation set does not improve in an echo, the learning rate is halved. The main results are shown in Table 3. Our model scores the best results. On ROUGE-1, ROUGE-2 and ROUGE-L, it can enhance about 0.28, 0.53, 0.26 on DUC-2004 and about 0.42, 0.33, 0.58 on Gigaword. Furthermore, we consider some examples of generating summarization (as shown in Table 2). Our model is very close to the real headline, especially in sentence 5, there is no difference between *gazprom and chevron set up a joint venture* and *gazprom chevron set up joint venture.*

6. Conclusion and future work

We propose an attention-based Tree-LSTM model for Sentence Summarization. Our model utilizes an attention-based syntactic structure for input to generate output. Block-alignment is used for the syntax blocks and inter-alignment is used inside the block pairs. Further, block-alignment can prevent structural deviations in the long sentences and the inter-alignment can increase the flexibility of the generation in the block. This is also an end-to-end summary generation model. Firstly, we block the input and output sentence-pairs according to the syntax tree. Then the blocks between the sentence-pairs uses block-alignment mode and inner-block between them uses inter-alignment mode. Finally we get a very good model. However, the experimental results rely heavily on the results of syntax tree generation. As a next step, on the basis of the sentence summary and syntax structure, we will research on the generation of paragraph-level and article-level summarization.

Table 3. A few sentence summaries produced on dataset

input sentences	Headline(**H**) and our Output(**O**)
S(1): ministers from the european union and its mediterranean neighbors gathered here under heavy security on monday for an unprecedented conference on economic and political cooperation	**H:** european mediterranean ministers gather for landmark conference by julie bradford
	O: european union and its mediterranean neighbors gathered for conference on economic and political cooperation .
S(2): the death toll from a school collapse in a haitian shanty-town rose to ## after rescue workers uncovered a classroom with ## dead students and their teacher , officials said saturday .	**H:** toll rises to ## in haiti unk : official
	O:officials said the death toll in haiti school rises to ##
S(3): the white house on thursday warned iran of possible new sanctions after the un nuclear watchdog reported that tehran had begun sensitive nuclear work at a key site in defiance of un resolutions .	**H:** us warns iran of step backward on nuclear issue
	O:us warns iran of possible new sanctions
S(4): an explosion in iraq 's restive northeastern province of diyala killed two us soldiers and wounded two more , the military reported monday .	**H:** two us soldiers killed in iraq blast december toll ##
	O:an explosion in diyala killed two us soldiers and wounded ##
S(5): russia 's gas and oil giant gazprom and us oil major chevron have set up a joint venture based in resource-rich northwestern siberia , the interfax news agency reported thursday quoting gazprom officials	**H:** gazprom chevron set up joint venture
	O:gazprom and chevron set up a joint venture

Acknowledgements

This work was supported by National Natural Science Foundation of China (number: 61373148, 61502151) and National Social Science Foundation (number: 12BXW040).

References

1. D. Andor, C. Alberti, D. Weiss, et al., "Globally normalized transition-based neural networks", *arXiv preprint arXiv: 1603.06042*, 2016.
2. Ayana, S. Shen, Z. Liu, et al., "Neural headline generation with minimum risk training", *arXiv preprint arXiv: 1604.01904*, 2016.
3. D. Bahdanau, K. Cho, Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv: 1409.0473,* 2014.
4. D. Chen, C.D. Manning, "A fast and accurate dependency parser using neural networks", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 740-750, 2014.
5. J. Clarke, M. Lapata, "Global inference for sentence compression: An integer linear programming approach", *Journal of Artificial Intelligence Research*, vol.31, pp. 399-429, 2008.
6. T. Cohn, M. Lapata, "Sentence compression beyond word deletion", in *Proceedings of the International Conference on Computational Linguistics*, pp. 137-144, 2008.
7. K. Filippova, Y. Altun, "Overcoming the lack of parallel data in sentence compression", *in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1481-1491, 2013.
8. K. Knight, D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression", *Artificial*

Intelligence, vol.139, no.1, pp. 91-107, 2002.

9. P. Koehn, H. Hoang, A. Birch et al., "Moses: Open source toolkit for statistical machine translation", in *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pp. 177-180, 2007.
10. K. Lopyrev, "Generating news headlines with recurrent neural networks", *arXiv preprint arXiv: 1512.01712*, 2015.
11. C. D. Manning, M. Surdeanu, J. Bauer, et al., "The stanford corenlp natural language processing toolkit", in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60, 2014.
12. C. D. Manning, P. Raghavan, "Introduction to information retrieval", *People's Posts and Telecommunications Press,* pp. 824-825, 2010
13. J. Nivre, "Algorithms for Deterministic Incremental Dependency Parsing", *Computational Linguistics,* vol.34, no.4, pp. 513-553, 2008.
14. A. M. Rush, S. Chopra, J. Weston, "A neural attention model for abstractive sentence summarization", *arXiv: 1509. 00685v2*, 2015.
15. K. S. Tai, R. Socher, C. D. Manning, "Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks", *arXiv preprint arXiv: 1503.00075*, 2015.
16. D. Weiss, C. Alberti, M. Collins, "Structured training for neural network transition-based parsing." *arXiv preprint arXiv: 1506.06158,* 2015.
17. K. Woodsend, Y. Feng, M. Lapata, "Generation with quasi-synchronous grammar", in *Proceedings of the Association for Computational Linguistics,* pp. 513-523, 2010.
18. S. Wubben, A. V. D Bosch, E. Krahmer, "Sentence simplification by monolingual machine translation", in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, vol.1, pp. 1015-1024, 2012.
19. D. Zajic, B. Dorr, R. Schwartz, "Bbn/umd at duc-2004: Topiary", in *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston,* pp. 112-119, 2004.
20. Y. Zhang, J. Nivre, "Transition-based dependency parsing with rich non-local features", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics,* pp. 740-750, 2011.
21. H. Zhou, Y. Zhang, S. Huang, et al., "A Neural Probabilistic Structured-Prediction Model for Transition-Based Dependency Parsing", *Association for Computational Linguistics,* vol. 1, pp. 1213-1222, 2015.

Wenfeng Liu is a Ph.D. candidate at the School of Information Science and Engineering, Shandong Normal University. He received the Bachelor degree in Computer Science from Ludong University, Yantai, China, in 2005, the Master degree in Computer Application Technology from University of Science and Technology Liaoning, Anshan, china, in 2011. And he is a lecturer at the department of Computer and Information, Heze University, china. His research interest covers natural language processing, machine learning and deep learning.

Peiyu Liu is a professor at the School of Information Science and Engineering, Shandong Normal University. His research interest covers NLP, sentiment analysis, topic detection and tracking, and network information security. He is also the corresponding author of this paper.

Yuzhen Yang received the Ph. D. at the School of Information Science and Engineering, Shandong Normal University. She is a Lecturer at the department of Computer and Information, Heze University, china. Her research interest covers natural language processing, machine learning and deep learning.

Yaling Gao is a Computer Network Engineer at Ruipu Peony Industrial Technology Development Co., Ltd, China, and received the grade 4 of National Computer Rank Examination Certificate in 2011. Her research interest covers computer network and natural language processing.

Jing Yi is a Ph. D. candidate at the School of Information Science and Engineering, Shandong Normal University. She is an associate professor at School of Computer Science and Technology, Shandong Jianzhu University, Jinan, China. Her research interest covers network security and data mining.