A Language Model based Evaluator for Sentence Compression

Yang Zhao **Zhiyuan Luo**
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
{zhao,zyluo24}@is.s.u-tokyo.ac.jp

Akiko Aizawa
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo
aizawa@nii.ac.jp

Abstract

We herein present a language-model-based evaluator for deletion-based sentence compression, and viewed this task as a series of deletion-and-evaluation operations using the evaluator. More specifically, the evaluator is a syntactic neural language model that is first built by learning the syntactic and structural collocation among words. Subsequently, a series of trial-and-error deletion operations are conducted on the source sentences via a reinforcement learning framework to obtain the best target compression. An empirical study shows that the proposed model can effectively generate more readable compression, comparable or superior to several strong baselines. Furthermore, we introduce a 200-sentence test set for a large-scale dataset, setting a new baseline for the future research.

1 Introduction

Deletion-based sentence compression aims to delete unnecessary words from source sentence to form a short sentence (compression) while retaining grammatical and faithful to the underlying meaning of the source sentence. Previous works used either machine-learning-based approach or syntactic-tree-based approaches to yield most readable and informative compression (Jing, 2000; Knight and Marcu, 2000; Clarke and Lapata, 2006; McDonald, 2006; Clarke and Lapata, 2008; Filippova and Strube, 2008; Berg-Kirkpatrick et al., 2011; Filippova et al., 2015; Bingel and Søgaard, 2016; Andor et al., 2016; Zhao et al., 2017; Wang et al., 2017). For example, (Clarke and Lapata, 2008) proposed a syntactic-tree-based method that considers the sentence compression task as an optimization problem by using integer linear programming, whereas (Filippova et al., 2015) viewed the sentence compression task as a sequence labeling problem using the recurrent neural network (RNN), using maximum likelihood as the objective function for optimization. The latter sets a relatively strong baseline by training the model on a large-scale parallel corpus. Although an RNN (e.g., Long short-term memory networks) can implicitly model syntactic information, it still produces ungrammatical sentences. We argue that this is because (i) the labels (or compressions) are automatically yielded by employing the syntactic-tree-pruning method. It thus contains some errors caused by syntactic tree parsing error, (ii) more importantly, the optimization objective of an RNN is the likelihood function that is based on individual words instead of readability (or informativeness) of the whole compressed sentence. A gap exists between optimization objective and evaluation. As such, we are of great interest that: (i) can we take the readability of the whole compressed sentence as a learning objective and (ii) can grammar errors be recovered through a language-model-based evaluator to yield compression with better quality?

To answer the above questions, a syntax-based neural language model is trained on large-scale datasets as a readability evaluator. The neural language model is supposed to learn the correct word collocations in terms of both syntax and semantics. Subsequently, we formulate the deletion-based sentence compression as a series of trial-and-error deletion operations through a reinforcement learning framework. The policy network performs either RETAIN or REMOVE action to form a compression, and receives a reward (e.g., readability score) to update the network.

The empirical study shows that the proposed method can produce more readable sentences that preserve the source sentences, comparable or superior to several strong baselines. In short, our contributions are two-fold: (i) an effective syntax-based evaluator is built as a post-hoc checker, yielding compression with better quality based upon the evaluation metrics; (ii) a large scale news dataset with 1.02 million sentence compression pairs are compiled for this task in addition to 200 manually created sentences. We made it publicly available.

2 Methodology

2.1 Task and Framework

Formally, deletion-based sentence compression translates word tokens, $(w_1, w_2, ..., w_n)$ into a series of ones and zeros, $(l_1, l_2, ..., l_n)$, where n refers to the length of the original sentence and $l_i \in \{0, 1\}$. Here, "1" refers to RETAIN and "0" refers to REMOVE. We first converted the word sequence into a dense vector representation through the parameter matrix E. Except for word embedding, $(e(w_1), e(w_2), ..., e(w_n))$, we also considered the part-of-speech tag and the dependency relation between w_i and its head word as extra features. Each part-of-speech tag was mapped into a vector representation, $(p(w_1), p(w_2), ..., p(w_n))$ through the parameter matrix P, while each dependency relation was mapped into a vector representation, $(d(w_1), d(w_2), ..., d(w_n))$ through the parameter matrix D. Three vector representations are concatenated, $[e(w_i); p(w_i); d(w_i)]$ as the input to the next part, policy network.

Figure 1 shows the graphical illustration of our model. The policy network is a bi-directional RNN that uses the input $[e(w_i); p(w_i); d(w_i)]$ and yields the hidden states in the forward direction, $(h_1^f, h_2^f, ..., h_n^f)$, and hidden states in the backward direction, $(h_1^b, h_2^b, ..., h_n^b)$. Then, concatenation of hidden states in both directions, $[h_i^f; h_i^b]$ are followed by a nonlinear layer to turn the output into a binary probability distribution, $y_i = \sigma(W[h_i^f; h_i^b])$ where σ is a nonlinear function $sigmoid$, and W is a parameter matrix.

The policy network continues to sample actions from the binary probability distribution above until the whole action sequence is yielded. In this task, binary actions space is {RETAIN, REMOVE}. We turn the action sequence into the predicted compression, $(w_1, w_2, ..., w_m)$, by deleting the words whose current action is REMOVE. Then

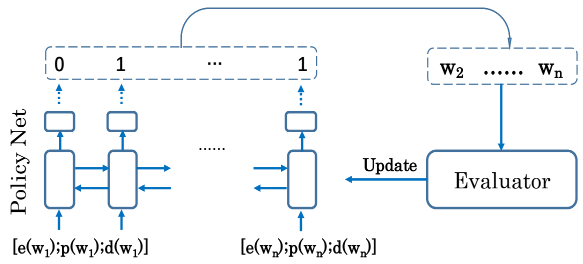

Figure 1: Graphical illustration of the framework.

the $(w_1, w_2, ..., w_m)$ is fed into a pre-trained evaluator which will be described in the next section.

2.2 Syntax-based Evaluator

The syntax-based evaluator should assess the degree to which the compressed sentence is grammatical, through being used as a reward function during the reinforcement learning phase. It needs to satisfy three conditions: (i) grammatical compressions should obtain a higher score than ungrammatical compressions, (ii) for two ungrammatical compressions, it should be able to discriminate them through the score despite the ungrammaticality, (iii) lack of important parts (such as the primary subject or verb) in the original sentence should receive a greater penalty.

We therefore considered an ad-hoc evaluator, i.e., the syntax-based language model (evaluator-SLM) for these requirements. It integrates the part-of-speech tags and the dependency relations in the input, while the output to be predicted is the next word token. We observed that the prediction of the next word could not only be based on the previous word but also the syntactic components, e.g., for the part-of-speech tag, the noun is often followed by a verb instead of an adjective or adverb and the integration of the part-of-speech tag allows the model to learn such correct word collocations. Figure 2 shows the graphical illustration of the evaluator-SLM where the input is $x_i = [e(w_i); p(w_i); d(w_i)]$, followed by a bi-directional RNN whose last layer is the Softmax layer used to represent word probability distribution. Similar to (Mousa and Schuller, 2017), we added two special tokens, <S> and </S> in the input so as to stagger the hidden vectors, thus avoiding self-prediction. Finally, we have the following formula as one part of the reward functions in the learning framework.

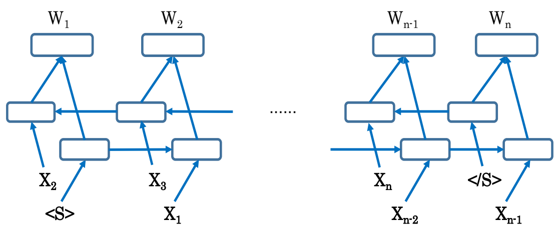

Figure 2: Graphical illustration of bi-directional recurrent neural network language model.

$$R_{SLM}(\widehat{Y}) = e^{(\frac{1}{|\widehat{Y}|}\sum_{t=1}^{|\widehat{Y}|} logP_{LM}(y_t|y_{0:t-1}))} \quad (1)$$

where $R_{SLM} \in [0,1]$ and $\widehat{Y}$ is the predicted compression by the policy network.

Further, it is noteworthy that the performance comparison should be based on a similar compression rate[1] (CR) (Napoles et al., 2011), and a smooth reward function $R_{CR} = \frac{(a+b)^{(a+b)}}{a^a b^b} x^a(1-x)^b$ (both a, b are positive integers; e.g. $a = 2$, $b = 2$ could lead the compression rate to 0.5) is also used to attain a compressed sentence of similar length.

The total reward is $R = R_{SLM} + R_{CR}$. By using policy gradient methods (Sutton et al., 2000), the policy network is updated with the following gradient:

$$\nabla\mathcal{L}(\theta) = \sum_{t=1}^{|\widehat{Y}|} R(\widehat{Y})\nabla log\pi_\theta(a_t|S_t) \quad (2)$$

Where $a_t \in$ {RETAIN, REMOVE}, is the action token by the policy network, and S_t refers to hidden state of the network, $[h_i^f; h_i^b]$ (section 2.1).

3 Experiments

3.1 Data

As neural network-based methods require a large amount of training data, we for the first time considered using Gigaword[2], a news domain corpus. More specifically, the first sentence and the headline of each article are extracted. After data cleansing, we finally compiled 1.02 million sentence and headline pairs (see details here[3]). It is noteworthy that the headline is not the extractive compression. Further, we asked two near native English speakers to create 200 extractive compressions for the first 200 sentences of this dataset; using it as the testing set, the first 1,000 sentences (excluding the testing set) is the development set, and the remainder is the training set. To assess the inter-assessor agreements, we computed Cohen 's unweighted κ. The computed unweighted κ was 0.423, reaching a moderate agreement level[4]

The second dataset we used was the Google dataset that contains 200,000 sentence compression pairs (Filippova et al., 2015). For the purpose of comparison, we used the very first 1,000 sentences as the testing set, the next 1,000 sentences as the development set, and the remainder as the training set.

3.2 Comparison Methods

We choose several strong baselines; the first one is the dependency-tree-based method that considers the sentence compression task as an optimization problem by using integer linear programming[5]. Inspired by (Filippova and Strube, 2008), (Clarke and Lapata, 2008), and (Wang et al., 2017), we defined some constrains: (1) if a word is retained in the compression, its parent should be also retained. (2) whether a word w_i is retained should partly depend on the word importance score that is the product of the TF-IDF score and headline score $h(w_i)$, $tf\text{-}idf(w_i) \cdot h(w_i)$ where $h(w_i)$ represents that whether a word (limited to nouns and verbs) is also in the headline. $h(w_i)$=5 if w_i is in the headline; $h(w_i)$=1 otherwise. (3) the dependency relations, *ROOT, dobj, nsubj, pobj*, should be retained as they are the skeletons of a sentence. (4) the sentence length should be over than α but less than β. (5) the depth of the node (word), $\lambda dep(w_i)$, in the dependency tree. (6) the word with the dependency relation *amod* is to be removed. It is noteworthy that the method is unsupervised.

The second method is the long short-term memory networks (LSTMs) which showed strong promise in sentence compression by (Filippova et al., 2015). The labels were obtained using the dependency tree pruning method (Filippova and Altun, 2013) and the LSTMs were applied in a supervised manner. Following their works, we also

[1]compression rate is the length of compression divided by the length of the sentence.

[2]https://catalog.ldc.upenn.edu/ldc2011t07

[3]https://github.com/code4conference/Data

[4](Landis and Koch, 1977) characterize κ values <0 as no agreement, 0 $\sim$ 0.20 as slight, 0.21 $\sim$ 0.40 as fair, 0.41 $\sim$ 0.60 as moderate, 0.61 $\sim$ 0.80 as substantial, and 0.81 $\sim$ 1 as almost perfect agreement.)

[5]we use http://pypi.python.org/pypi/PuLP

Gigaword Dataset	Annotator 1		Annotator 2		
	F_1	**RASP-F_1**	F_1	**RASP-F_1**	**CR**
#1 Seq2seq with attention	54.9	60.3	58.6	64.6	0.53
#2 Dependency tree+ILP	58.0	65.1	61.0	70.9	0.55
#3 LSTMs+pseudo label	60.3	64.1	64.1	69.2	0.51
#4 Evaluator-LM	64.5	67.3	66.9	72.2	0.50
#5 Evaluator-SLM	**65.0**	**69.6**	**68.2**	**73.9**	0.51

Table 1: F_1 and RASP-F_1 results for Gigaword dataset.

consider the labels yielded by our dependency-tree-based method as pseudo labels and employ LSTMs as a baseline.

Furthermore, for a comprehensive comparison, we applied the sequence-to-sequence with attention method widely used in abstractive text summarization for sentence compression. Previous works such as (Rush et al., 2015; Chopra et al., 2016) have shown promising results with this framework, although the focus was generation-based summarization rather than extractive summarization. More specifically, the source sequence of this framework is the original sentence, while the target sequence is a series of zeros and ones (zeros represents REMOVE and ones represents RETAIN). Further, we incorporated dependency labels and part-of-speech tag features in the source side of the sequence-to-sequence method.

3.3 Training

The embedding size for word, part-of-speech tag, and the dependency relation is 128. We employed the vanilla RNN with a hidden size of 512 for both the policy network and neural language model. The mini-batch size was chosen from $[5, 50, 100]$. Vocabulary size was 50,000. The learning rate for neural language model is 2.5e-4, and 1e-05 for the policy network. For policy learning, we used the REINFORCE algorithm (Williams, 1992) to update the parameters of the policy network and find an policy that maximizes the reward. Because starting from a random policy is impractical owing to the high variance, we pre-trained the policy network using pseudo labels in a supervised manner. For the comparison methods, the hyperparameters and were set to 0.4 and 0.7, respectively, and was set to 0.5. For reproduction, we released the source code here[6].

[6]https://github.com/code4conference/code4sc

4 Result and Discussion

This section demonstrates the experimental results on both datasets. As the Gigaword dataset has no ground truth, we evaluated the baseline and our method on the 200-sentence test sets created by two human annotators. For the automatic evaluation, we employed F_1 and RASP-F_1 (Briscoe and Carroll, 2002) to measure the performances. The latter compares grammatical relations (such as ncsubj and dobj) found in the system compressions with those found in the gold standard, providing a means to measure the semantic aspects of the compression quality. For the human evaluation, we asked two near native English speakers to assess the quality of 50 compressed sentences out of the 200-sentence test set in terms of readability and informativeness. Here are our observations:

Gigaword	*Readability*	*Informativeness*
\$1 LSTMs	3.56	3.10
\$2 SLM	**4.16**†	3.16

Table 2: Human Evaluation for Gigaword dataset. †stands for significant difference with 0.95 confidence in the column.

Google Dataset	F_1	**RASP-F_1**	**CR**
&1 Seq2seq with attention	71.7	63.8	0.34
&2 LSTM (Filippova, 2015)	82.0	-	0.38
&3 LSTMs (our implement)	84.8	81.9	0.40
&4 Evaluator-LM	85.0	82.0	0.41
&5 Evaluator-SLM	85.1	82.3	0.39

Table 3: F_1 and RASP-F_1 results for Google dataset.

(1) As shown in Table 1, our Evaluator-SLM-based method yields a large improvement over the baselines, demonstrating that the language-model-based evaluator is effective as a post-hoc grammar checker for the compressed sentences. This is also validated by the significant improvement in the readability score in Table 2 (\$1 vs \$2). To investigate the evaluator in detail, a case study is shown in section 4.1.

SENTENCE	The	Dalian	shipyard	has	built	two	new	huge	ships	$e^{-\mathrm{logR}}$
POS tags	*DET*	*ADJ*	*NOUN*	*VERB*	*VERB*	*NUM*	*ADJ*	*ADJ*	*NOUN*	
DEP. rels	*det*	*compound*	***nsubj***	*aux*	***root***	*nummod*	*amod*	*amod*	***dobj***	
#1	The	Dalian	shipyard	has	built	two	new	huge	ships	59.8
#2	The	Dalian	shipyard	has	built	two	new	huge		140.5
#3	The	Dalian	shipyard	has		two	new	huge	ships	582.9
#4	The	Dalian		has	built	two	new	huge	ships	1313.5
#5	The	Dalian		has	built	two			ships	1244.8
#6	The			has	built	two			ships	1331.2
#7	The	Dalian	shipyard	has	built	two		huge	ships	46.9
#8	The	Dalian	shipyard	has	built	two			ships	18.2
#9	The		shipyard	has	built	two			ships	66.5

Figure 3: Case study for evaluator.

(2) by comparing annotator 1 with annotator 2 in Table 1, we observed different performances for two annotated test sets, showing that compressing a text while preserving the original sentence is subjective across the annotators.

(3) As for Google news dataset, LSTMs (LSTM+pos+dep) (&3) is a relatively strong baseline, suggesting that incorporating dependency relations and part-of-speech tags may help model learn the syntactic relations and thus make a better prediction. When further applying Evaluator-SLM, only a tiny improvement is observed (&3 vs &4), not comparable to the improvement between #3 and #5. This may be due to the difference in perplexity of the our Evaluator-SLM. For Gigaword dataset with 1.02 million instances, the perplexity of the language model is 20.3, while for the Google news dataset with 0.2 million instances, the perplexity is 76.5.

(4) To further explore the degree to which syntactic knowledge (dependency relations and part-of-speech tags) is helpful to evaluator (language model), we implemented a naive language model, i.e., Evaluator-LM, which did not include dependency relations and part-of-speech tags as input features. The results shows that small improvements are observed on two datasets (#4 vs #5; &4 vs &5), suggesting that incorporating syntactic knowledge may help evaluator to encourage more unseen but reasonable word collocations.

4.1 Evaluator Analysis

To further analyze the Evaluator-SLM performance, we used an example sentence, “The Dalian shipyard has built two new huge ships” to observe how a language model scores different word deletion operations. We converted the reward function R_{SLM} to $e^{-logR_{SLM}}$ for a better observation (similar to ”sentence perplexity”, the higher the score is, the worse is the sentence). As shown in Figure 3, deleting the object(#2), verb(#3), or subject(#4) results in a significant increase in ”sentence perplexity”, implying that the syntax-based language model is highly sensitive to the lack of such syntactic components. Interestingly, when deleting words such as new or/and huge, the score becomes lower, suggesting that the model may prefer short sentences, with unnecessary parts such as *amod* being removed. This property makes it quite suitable for the sentence compression task aiming to shorten sentences by removing unnecessary words.

5 Conclusion

We presented a syntax-based language model for the sentence compression task. We employed unsupervised methods to yield labels to train a policy network in a supervised manner. The experimental results demonstrates that the compression could be further improved by a post-hoc language-model-based evaluator, and our evaluator-enhanced model performs better or comparable upon the evaluation metrics on two large-scale datasets.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP15H01721, JP18H03297. We are thankful for the reviewers’ helpful comments and suggestions. We would also like to thank Filippova for sharing their data with us and Clarke for the Annotator Sentence Compression Instructions in his PhD dissertation.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 2442–2452.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 481–490.

Joachim Bingel and Anders Søgaard. 2016. Text simplification as tree labeling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 337–343.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 93–98.

James Clarke and Mirella Lapata. 2006. Constraint-based sentence compression an integer programming approach. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pages 144–151.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 360–368.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. *Fifth International Natural Language Generation Conference on - INLG '08* page 25.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics, pages 310–315.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, pages 703–710.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* pages 159–174.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 1023–1032.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. pages 91–97.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 379–389.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. pages 1057–1063.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an lstm-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1385–1393.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, Springer, pages 5–32.

Yang Zhao, Hajime Senuma, Xiaoyu Shen, and Akiko Aizawa. 2017. Gated neural network for sentence compression using linguistic knowledge. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 480–491.