How much can Syntax help Sentence Compression ?

Hoa Le, Christophe Cerisara, Claire Gardent

How much can Syntax help Sentence Compression ?

Hoa T. Le[1], Christophe Cerisara, Claire Gardent[2]

[1] Laboratory LORIA Nancy, France
[2] CNRS/LORIA Nancy, France
{hoa.le, christophe.cerisara, claire.gardent}@loria.fr

Abstract. Sentence compression involves selecting key information present in the input and rewriting this information into a short, coherent text. Using the Gigaword corpus, we provide a detailed investigation of how syntax can help guide both extractive and abstractive sentence compression. We explore different ways of selecting subtrees from the dependency structure of the input sentence; compare the results of various models and show that preselecting information based on syntax yields promising results.

1 Introduction

Interestingly, while previous work on sentence compression has often focused on extractive compression i.e., compressions where most of the words occurring in the short sentence version are also present in the corresponding input [5, 6], the Gigaword corpus can be viewed as a corpus containing both extractive and abstractive compressions (or sentence summarization).

Previous work on this dataset has used various ways of selecting key information in the input sentence. [3] uses dependency subtrees and information extraction triples to enrich the input of a encoder-decoder model. [1] investigates the use of linked entities to guide the decoder. [19] propose an encoding model which includes a gate network to select key information from the input sentence. [16] propose to enrich the encoder with information about the syntactic structure of the input sentence. [2] use target summaries as soft templates to guide the sequence-to-sequence model.

In this paper, we provide a detailed investigation of how *syntax* can help guide sentence compression. We explore different ways of selecting subtrees from the dependency structure of the input sentence. We evaluate the results on the Gigaword corpus using an oracle setting and distinguishing between extractive and abstractive corpus instances. And we show that preselecting information based on syntax yields promising results.

2 Related Work

Sequence-to-sequence models nowadays are a popular model used for summarization task but it's still far from perfect. A lot of its problems have been noticed

in the community. [1, 3, 10, 11, 16] observed that sequence models can produce incorrect, hallucinated and non-factual output. One common remedy often used is integrating additional structural bias to make sure that the attention of the model spreads over key information in the source. [1] observed that information around entities could be built up the topic of the summary. They proposed to associate with a linked entities topic module to guide the decoding process. [3] used open information extraction and dependency parsing technique to achieve actual facts from source text and force the generator to respect these descriptions. In the same spirit, [16] explored the use of syntactic and relation from constituency parsing, [10] employed TextRank algorithm and [11] relied on entailment relations.

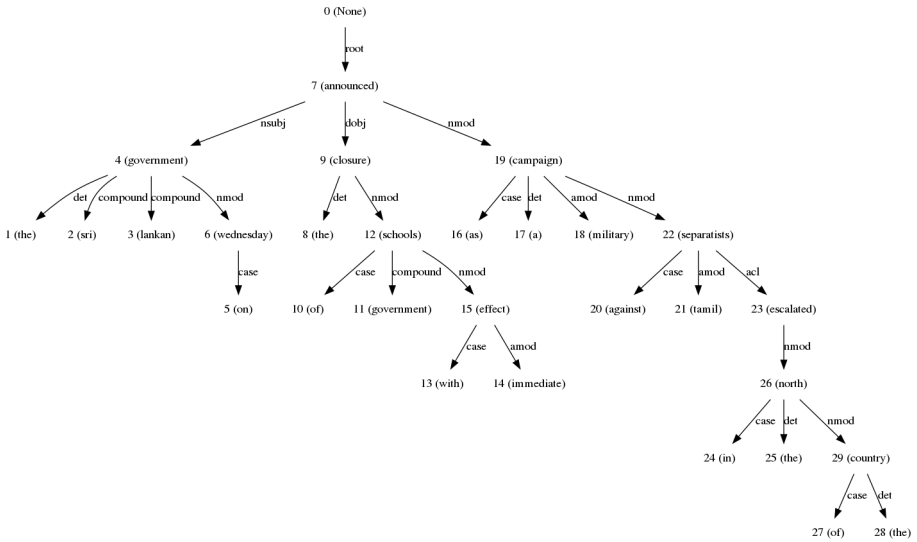
In parrallel, another research path is to directly learn to distill out important sentences from the source document. This is applied specifically on CNN/Daily Mail dataset. [13] proposed to cast extraction as a ranking problem and used reinforcement learning to directly optimize the final output. They argued that cross-entropy loss is not suitable metric on this task. However, this framework lacks of a rewriting component. Mimicking how human summarizes text, [4] proposed both components sentence extraction and rewriting network in a unified model and trained an agent to globally optimize them in an end-to-end manner.

Previous studies on summarization tasks, especially on Gigaword dataset, have only used dependency parsing as additional structural bias to the models [3, 10, 16]. Subtrees from dependency structure are still not properly explored to help narrowing down input sentence into a short, concise and less ambiguous piece of information. We investigate different ways of selecting subtrees to help decoder to rewrite better.

3 Extraction of Dependency Subtrees

We investigate different strategies for the extraction of the subtrees from the dependency structure of the input sentences. In all cases, we start from "sentence subtrees" i.e., subtrees rooted in a node which has an "nsubj" or an "nsubjpass" child node. Given such sentence trees, we then extract the following subtrees:

- **S-Tree**: All sentence trees (as just defined).
- **1L:1R**: all subtrees of the sentence tree which contain one left- and one right-child. E.g., given the example in Figure 1, *"government announced closure"* and *"government announced campaign"*
- **1L:AR (AL:1R)**: all subtrees of the sentence tree which contain one left-(right-) child and all right- (left-) children. E.g., *"government announced closure campaign"*
- **Auxl**: All intermediate subtrees below a level from subtree depth 1 containing "nsubj" or "nsubjpass".



S-tree, Depth 1	government anounced closure campaign
S-tree, Depth 2	sri lankan government on wednesday anounced schools closure military campaign separatists
1L:1R, Depth 1	government anounced closure government anounced campaign
1L:AR, Auxl, Depth 1	sri lankan government on wednesday anounced closure campaign

Fig. 1. Dependency Tree and example extracted subtress for the sentence *"The Sri-Lankan government on Wednesday anounced the closure of government schools with immediate effect as a military campaign against Tamil separatists escalated in the North of the country"* .

We ignore all children node whose parent dependency relation is "punct" or "det" as we observe that sentence compressions in the dataset are usually short and very concise. For each subtree type, we recursively extract subtrees of depth 1, 2 and 3. Figure 1 shows some example (linearisation of) extracted subtrees.

4 Model

Following [4], we use a two-steps model which combines an extractor agent to select dependency subtrees and an abstractor network to rewrite the extracted subtrees. The two networks are connected using reinforcement learning to overcome the non-differentiable behavior of the extractor and optimise with respect to the ROUGE evaluation, a standard metric used for sentence compression.

4.1 Extractor Network

First, a temporal convolutional network [8] is used to compute representation r_j of each individual subtree in the input and then a bidirectional LSTM [7] is applied on these convolutional output to obtain a stronger representation, denoted as h_j. Next, to select the most salient subtrees from the input sentence, a Pointer Network [17] is employed to get the extraction probability:

$$u_j^t = \begin{cases} v_p^\top \tanh(W_{p1}h_j + W_{p2}e_t) & \text{if } j_t \neq j_k, \forall k < t \\ -\infty & \text{otherwise} \end{cases} \tag{1}$$

$$P(j_t|j_1, \ldots, j_{t-1}) = \text{softmax}(u^t) \tag{2}$$

where e_t's are the output of the *glimpse* operation:

$$a_j^t = v_g^\top \tanh(W_{g1}h_j + W_{g2}z_t) \tag{3}$$

$$\alpha^t = \text{softmax}(a^t) \tag{4}$$

$$e_t = \sum_j \alpha_j^t W_{g1}h_j \tag{5}$$

In Eqn. 3, z_t is the output of the added LSTM decoder. All the W's and v's are trainable parameters. Similar to [4], we pretrain this extractor via a 'proxy' target label. For each ground-truth summary sentence, we find the most similar subtree via ROUGE-L$_{recall}$ metric and minimize this classification cross-entropy loss.

4.2 Abstractor Network

To generate compression, we use state-of-the-art sequence-to-sequence model with copy mechanism [15]. We also pretrain abstractor by taking pair of each summary and its extracted subtree (in section 4.1). The network is trained as usual on decoder language model $L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} \log P_{\theta_{abs}}(w_m|w_{1:m-1})$.

4.3 Reinforce Extraction

To make an extraction agent, we use vanilla policy gradient algorithm REINFORCE [18]. At each extraction step t, the agent observes the current state $c_t = (D, d_{j_{t-1}})$, where $d \in D$: set of document sentence input. It samples an action $j_t \sim \pi_{\theta_a,\omega}(c_t, j) = P(j)$ from Eqn. 2 to extract a subtree and receive a reward. We denote trainable parameters of the extractor agent as $\theta = \{\theta_a, \omega\}$ (in section 4.1). Then, because vanilla REINFORCE yields high variance, we maximize this following policy gradient objective:

$$\nabla_{\theta_a,\omega} J(\theta_a, \omega) = \mathbb{E}[\nabla_{\theta_a,\omega} \log \pi_\theta(c, j) A^{\pi_\theta}(c, j)] \tag{6}$$

where $A^{\pi_\theta}(c, j)$ is the *advantage function*, calculated as: $A^{\pi_\theta}(c, j) = Q^{\pi_{\theta_a,\omega}}(c, j) - b_{\theta_c,\omega}(c)$. As we can see here, the total return R_t could be used as an estimate of action-value function $Q(c_t, j_t)$, a baseline $b_{\theta_c,\omega}(c)$ is needed to reduce its variance. Finally, the baseline is then also updated by minimizing this square loss: $L_c(\theta_c, \omega) = (b_{\theta_c,\omega}(c_t) - R_t)^2$.

4.4 Oracle

As the number of dependency subtrees extracted by our various strategies (cf. Section 3) can be very large, [4]'s two steps approach actually fails to learn a good selection strategy and underperforms.

To assess the degree to which syntax can help sentence compression, we therefore consider the results of an oracle setting in which we apply the abstractor of the Select-and-Paraphrase Model [4] to each of the subtrees produced for the input sentence by the various extraction strategies described in Section 3 and report the best ROUGE scores obtained.

We also compare our approach with an approach where the key information extracted from the input is the set of subject-predicate-object triples extracted from the input sentence by Stanford CoreNLP OpenIE tool.

5 Experiments

5.1 Data

We evaluate our approach on the Gigaword corpus [14], a corpus of 3.8M sentence-headline pairs and where the average input sentence length is 31.4 words (in the training corpus) and the average sentence compression length is 8.3 words. The test set consists of 1951 sentence/compression pairs. Like [14], we use 2000 sample pairs (among 189K pairs) as development set.

5.2 Extractive vs. Abstractive Compression

To better assess the impact of our approach on abstractive vs extractive compression, we divide the data (training, dev and test) into two parts: a part (extractive) where 80% of the tokens present in the sentence compression are present in the input and another part (abstractive) which contains all other instances. According to that criteria, out of the 1951 test instances, 207 are extractive and 1744 abstractive.

5.3 Evaluation Metric

We adopt ROUGE [12] for automatic evaluation. It measures the quality of summary by computing overlapping lexical units between the candidate summary and actual summaries. We report ROUGE-1 (unigram), ROUGE-2 (bi-gram) and ROUGE-L (LCS) F1 scores. ROUGE-1 and ROUGE-2 mainly represent informativeness while ROUGE-L is supposed to be assess the readability.

5.4 Hyperparameter Details

We use Adam optimizer [9] with learning rate 0.001 for extractor, abstractor ML and 0.0001 for extractor RL training. Vocabulary size of 30k, batch size 32 samples, gradient clipping of 2.0 and regularizing by early-stopping. For CNN,

we use 100 hidden units and 256 hidden units for all LSTM-RNNs. We truncate maximum length of input sentences to 100 tokens and target sentences to 30 tokens. ROUGE-recall is used to create proxy label data as we want extracted sentence to contain as much information as possible for paraphrasing. However, ROUGE-F_1 is used as reward for reinforce agent as the generation should be as *concise* as the gold target sentence.

6 Results

6.1 Full Select-and-Paraphrase Model

Table 1. Baseline (Seq2Seq trained on Sentence/Compression Pairs) vs. RL Select-and-Paraphrase Model (trained on S-Tree Data).

Model	**Extractive Data**			**Abstractive Data**		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Baseline	59.57	31.28	57.87	27.55	10.16	25.94
S&P Model	54.38	28.13	52.42	24.48	8.49	23.15

Table 1 shows the results comparing the baseline model (a simple seq2seq model trained on input sentence/compression pairs) and the full Select-and-Paraphrase Reinforcement (RL) learning model where the subtrees the extractor is trained on, are the S-trees. The RL model under-performs which indicates that the extractor does not succeed in selecting from among the dependency subtrees, the key information necessary for generating the sentence compression. It should be noted that the set of dependency subtrees can be very large (up to several hundreds) making it difficult to learn a good ranker for the extractor. Making the problem more tractable would require finding a method for limiting the number of dependency subtrees to be taken into consideration. We leave this point for further research and concentrate instead in the following section on analysing the impact on performance of the different extraction strategies described in Section 3.

6.2 Oracle Setting

Table 2 compares the baseline with an oracle abstracted trained on OpenIE triples and on different sets of dependency subtrees extracted from the parse tree of the input sentence.

OpenIE triples vs. Dependency Subtrees. One first observation is that scores are lower when taking as input OpenIE triples rather than Dependency subtrees. The results show that our specific S-tree heuristic rule outperforms arbitrary OpenIE triples by 9, 7, 9 rouge-1,-2,-L point respectively on oracle setup for the extractive

Table 2. Baseline vs. Oracle Results.

Models	Extractive Data			Abstractive Data		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Baseline	59.57	31.28	57.87	27.55	10.16	25.94
Oracle						
OpenIE	51.21	24.1	48.7	27.35	9.24	25.68
S-Tree	60.52	31.21	57.29	30.61	10.69	28.77
1L:1R	46.33	19.15	43.6	22.63	5.84	21.03
1L:1R, Auxl	64.04	28.32	60.55	33.23	10.01	30.3
1L:AR, AL:1R	43.99	20.4	42	21.16	5.48	19.71
1L:AR, AL:1R, Auxl	62.57	32.02	58.98	30.61	10.69	28.77
S-Tree, 1L:1R, Auxl	68.95	36.34	65.49	38.95	13.9	35.54
S-Tree, 1L:1R, Auxl, 1L:AR, AL:1R, Auxl	70.38	38.79	66.4	40	14.75	36.34

data and 3, 1 and 3 point for the abstractive data. OpenIE triples were in fact used by [3] on the same task and same dataset to improve faithfulness i.e., to favour output which is semantically correct with respect to the input sentence. As [3] achieved good scores on the Gigaword data and S-trees outperforms OpenIE triples, this suggests that S-Tree subtrees could help further improving results provided a good extractor can be developed.

Extractive vs. Abstractive. Unsurprisingly, the impact of the input dependency subtrees is much larger on extractive data. For extractive compressions, the scores increase by roughly a factor of two suggesting that the match between input dependency subtrees and compression is much larger for extractive compressions. This is in line with previous work [5, 6] which shows that extractive compressions can be found by searching in the parse tree of the input sentence for a spanning tree linking the content words of the compression. It also indicates that further improvements on the Gigaword dataset will require a better modelling of the paraphrases and variations occurring in abstractive compressions.

Auxiliary subtrees. We observe a large, significant delta between (1L:1R) and (1L:1R, Auxl) and similarly, between (1L:AR, AL:1R) and (1L:AR, AL:1R, Auxl). In fact, this increase shows up systematically in the experiments. That is, when considering subtrees below subject level, scores increase indicating that dependents and modifiers of these nodes often contain key information that is preserved in the compressed sentence. Gradual combination of each auxiliary subtrees with its higher level (1L:1R) or (1L:AR, AL:1R) can yield the most flexible phrase structure.

Syntax helps. The combination of subtrees shows substantial improvement. Among all possible setup, the (S-Tree, 1L:1R, Auxl, 1L:AR, AL:1R, Auxl) combination obtains the best performance. It is 10, 7, 9 rouge-1,-2-L points respectively higher than the first heuristic rule and the baseline seq2seq model on the extractive set. On the abstractive set, it is 13, 4, 11 rouge-1,-2,-L points higher respectively.

Table 3. Example of oracle and full source generation.

Source
fred west told the **truth** – and should be believed – **when he exonerated his wife** in the **murders** of ## young women before killing himself , a jury heard wednesday .
Subtrees
...
12. fred west believed when exonerated wife heard wednesday
13. fred west believed when exonerated murders heard wednesday
14. fred west believed when exonerated killing heard wednesday
15. fred west believed he exonerated wife heard wednesday
16. fred west believed he exonerated murders heard wednesday
17. fred west believed he exonerated killing heard wednesday
18. fred west believed a jury heard wednesday
...
Abstract
fred west told truth when he exonerated his wife of murder defense by unk unk
Full source generation
jury hears west tells truth to be believed to be believed
Subtrees generation
...
12. fred west says he 's exonerated
13. fred west says west nile murders
14. fred west says it was exonerated in killing of ##
15. fred west says he exonerated wife
16. fred west says he exonerated murders
17. fred west says he exonerated killing of killing
18. fred west s west virginia jury hears
...
Oracle
fred west says he exonerated wife (from subtree 15)

Qualitative analysis. Table 3 shows an example of multiple subtrees retrieved from input document and their generations. We can see that normal sequence-to-sequence with attention and copy mechanism struggles to identify important information and produces loops, repetition in the end. On the other hand, thanks to dependency structure, subtrees generation contain short, coherent and right-to-the-point message in the gold abstract.

7 Conclusion

We have provided an analysis of how various syntactic strategies impact automatic compression and shown that inputing sentence compression with subtrees extracted from the dependency parse of the input sentence can improve results. To improve the end-to-end RL compression model described in Section 4 so that it can benefit from such syntactic input, we are currently working on identifying

ways of restricting the number of subtrees to be considered and more generally, on improving the extractor.

References

1. Amplayo, R.K., Lim, S., Hwang, S.w.: Entity commonsense representation for neural abstractive summarization. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 697–707. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-1064, http://aclweb.org/anthology/N18-1064
2. Cao, Z., Li, W., Li, S., Wei, F.: Retrieve, rerank and rewrite: Soft template based neural summarization. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 152–161 (2018)
3. Cao, Z., Wei, F., Li, W., Li, S.: Faithful to the original: Fact aware neural abstractive summarization. CoRR **abs/1711.04434** (2017), http://arxiv.org/abs/1711.04434
4. Chen, Y.C., Bansal, M.: Fast abstractive summarization with reinforce-selected sentence rewriting. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 675–686. Association for Computational Linguistics (2018), http://aclweb.org/anthology/P18-1063
5. Filippova, K., Alfonseca, E., Colmenares, C.A., Kaiser, L., Vinyals, O.: Sentence compression by deletion with lstms. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 360–368. Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/D15-1042, http://aclweb.org/anthology/D15-1042
6. Filippova, K., Altun, Y.: 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. Association for Computational Linguistics (2013), http://aclweb.org/anthology/D13-1155
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/D14-1181, http://aclweb.org/anthology/D14-1181
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Li, C., Xu, W., Li, S., Gao, S.: Guiding generation for abstractive text summarization based on key information guide network. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). pp. 55–60. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-2009, http://aclweb.org/anthology/N18-2009
11. Li, H., Zhu, J., Zhang, J., Zong, C.: Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 1430–1441. Association for Computational Linguistics (2018), http://aclweb.org/anthology/C18-1121

12. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out (2004)
13. Narayan, S., Cohen, S.B., Lapata, M.: Ranking sentences for extractive summarization with reinforcement learning. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1747–1759. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-1158, http://aclweb.org/anthology/N18-1158
14. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 379–389. Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/D15-1044, http://aclweb.org/anthology/D15-1044
15. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1073–1083. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-1099, http://aclweb.org/anthology/P17-1099
16. Song, K., Zhao, L., Liu, F.: Structure-infused copy mechanisms for abstractive summarization. CoRR **abs/1806.05658** (2018), http://arxiv.org/abs/1806.05658
17. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2. pp. 2692–2700. NIPS'15, MIT Press, Cambridge, MA, USA (2015), http://dl.acm.org/citation.cfm?id=2969442.2969540
18. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: Machine Learning. pp. 229–256 (1992)
19. Zhou, Q., Yang, N., Wei, F., Zhou, M.: Selective encoding for abstractive sentence summarization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1095–1104. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-1101, http://aclweb.org/anthology/P17-1101