RL extraction of syntax-based chunks for sentence compression

Hoa T Le, Christophe Cerisara, Claire Gardent

RL extraction of syntax-based chunks for 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. While dependency parses have often been used for this purpose, we propose to exploit such syntactic information within a modern reinforcement learning-based extraction model. Furthermore, compared to other approaches that include syntactic features into deep learning models, we design a model that has better explainability properties and is flexible enough to support various shallow syntactic parsing modules. More specifically, we linearize the syntactic tree into the form of overlapping text segments, which are then selected with reinforcement learning and regenerated into a compressed form. Hence, despite relying on extractive components, our model is also able to handle abstractive summarization. We explore different ways of selecting subtrees from the dependency structure of the input sentence and compare the results of various models on the Gigaword corpus.

1 Introduction

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 work, we propose a model that exploits syntactic parsing to extract coherent segments from the source document, then selects the best of these segments with reinforcement learning, and finally regenerates the summary with a recent sequence-to-sequence model. This model can thus transparently handles

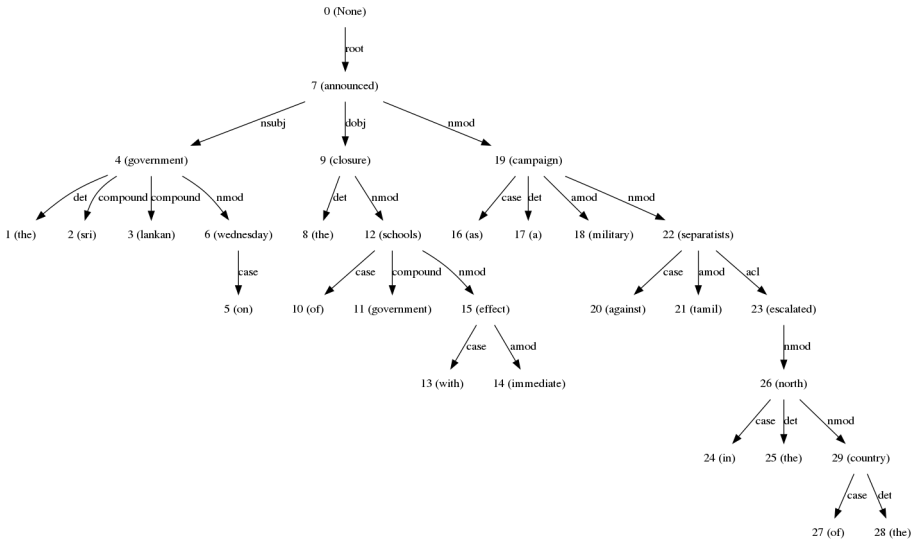
both types of extractive and abstractive summarizations. Furthermore, our approach departs from the recent end-to-end deep learning architectures by deterministically linearizing the syntactic tree into overlapping text segments to select with reinforcement learning. Although this method prevents a joint global optimization of the models, it also gives interesting adaptability and explainability properties. Adaptability, because it is easy to replace the syntactic parsing module with another shallow component, such as chunking, when full dependency parses are not available or not reliable; Explainability, because the segments that lead to the best summaries are clearly identified, which is not always the case with other deep learning approaches, for instance based on attention.

2 Related Work

Sequence-to-sequence models nowadays are a popular model used for summarization task but are still far from perfect. For instance, [1, 3, 10, 11, 16] observed that sequence models can produce incorrect, hallucinated and non-factual output. A common remedy often used consists to integrate 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 are related to the topic of the summary. They proposed to associate with linked entities a topic module to guide the decoding process. [3] used open information extraction and dependency parsing technique to infer actual facts from source text and force the generator to respect these descriptions. In the same spirit, [16] explored the use of syntactic relations from constituency parsing, [10] employed TextRank algorithm and [11] relied on entailment relations.

In parallel, another research path aims at directly learning to distill out important sentences from the source document. This is applied specifically on the CNN/Daily Mail dataset. [13] proposed to cast extraction as a ranking problem and used reinforcement learning to optimize the final objective. They argued that cross-entropy loss is not a suitable metric on this task. However, this framework lacks of a rewriting component. Mimicking how human summarizes text, [4] proposed both sentence extraction and rewriting components 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 the 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.



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 subtrees extracted from 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"* .

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 which root node has an "nsubj" or an "nsubjpass" child node. Given such sentence trees, we then extract the following subtrees:

- **S-Tree**: All sentence trees.
- **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 subtrees below a subtree at depth 1 that contains "nsubj" or "nsubjpass".

We ignore all children nodes whose parent dependency relation is "punct" or "det". For each subtree type, we recursively extract subtrees of depth 1, 2 and 3. Figure 1 shows some examples of linearized extracted subtrees.

4 Model

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

4.1 Extractor Network

Every linearized subtree from Section 3 is scored by the extractor network; this set of scored subtrees will later on be explored by the Reinforcement Learning agent to select the best candidates for summarization. In details, every subtree is passed to a temporal convolutional network [8] followed by a bidirectional LSTM [7] to produce a subtree embedding h_j. Assuming that h_j is selected, it is passed to an LSTM decoder for generation, which outputs z_t. A pointer network [17] computes a first attention e_t for z_t over all inputs $(h_j)_j$ with:

$$a_j^t = v_g^\top \tanh(W_{g1} h_j + W_{g2} z_t) \tag{1}$$

$$\alpha^t = \text{softmax}(a^t) \tag{2}$$

$$e_t = \sum_j \alpha_j^t W_{g1} h_j \tag{3}$$

It then computes the extraction probability of h_j with another attention:

$$u_j^t = v_p^\top \tanh(W_{p1} h_j + W_{p2} e_t)$$

$$P(j_t | j_1, \ldots, j_{t-1}) = \text{softmax}(u^t) \tag{4}$$

All the W's and v's are trainable parameters. Similarly to [4], we pretrain this extractor via a 'proxy' target label: for every 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. 4 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:

$$\nabla_{\theta_a,\omega} J(\theta_a, \omega) = \mathbb{E}[\nabla_{\theta_a,\omega} \log \pi_\theta(c, j) A^{\pi_\theta}(c, j)] \tag{5}$$

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: $L_c(\theta_c, \omega) = (b_{\theta_c,\omega}(c_t) - R_t)^2$.

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. We also report the ROUGE metrics over the whole corpus, to allow for comparison with related works.

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 rather capture readability.

5.4 Hyperparameter Details

We use the Adam optimizer [9] with learning rate 0.001 for cross-entropy training of the extractor and abstractor. We use a learning rate of 0.0001 for extractor RL training. The vocabulary size is 30k, the batch size 32 samples, we use gradient clipping of 2.0 and early-stopping. We use 256 hidden units for all LSTM-RNNs. We truncate the 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 sentences 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 shows the results comparing the baseline model (a seq2seq model trained on input sentence/compression pairs) and our full Select-and-Paraphrase Reinforcement Learning (RL) model where the extractor is trained on all sentence trees (S-trees). The RL model under-performs because the extractor does not manage to handle the large number of candidate sub-trees (up to several hundreds). We thus explore next, using an oracle RL selector, which of the selection methods proposed in Section 3 help to reduce the set of candidate sub-trees while still preserving the relevant information from the input document.

We also report in Table 1 the performances obtained with state-of-the-art summarization systems. These figures come from [2], which appears to be the best system on the Gigaword corpus reported in `http://nlpprogress.com/english/summarization.html`, as of May 2019.

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**			**Whole corpus**		
	R-1	**R-2**	**R-L**	**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	30.95	12.40	29.33
S&P Model	54.38	28.13	52.42	24.48	8.49	23.15	27.65	10.57	26.26
[2]							37.04	19.03	34.46

6.2 Oracle Setting

In Table 2, we compare our model with an "oracle reinforcement learning" component that always chooses the best candidate subtree to pass to the abstractor. This allows us to study the impact of each sub-tree selection processes described

in Section 3 and identify the ones that preserve relevant information to summarize.

We also apply our approach with another shallow syntactic process, by replacing the dependency parser by the CoreNLP OpenIE tool [3], which extracts a set of subject-predicate-object triples.

Table 2. Baseline vs. Oracle Results. The last row S-tree+ includes Stree, 1L:1R, Auxl, 1L:AR, AL:1R, Auxl

Model	**Extractive Data**			**Abstractive Data**			**Whole corpus**		
	R-1	**R-2**	**R-L**	**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	30.95	12.40	29.33
Oracle									
OpenIE	51.21	24.1	48.7	27.35	9.24	25.68	29.88	10.82	28.12
S-Tree	60.52	31.21	57.29	30.61	10.69	28.77	33.78	12.88	31.80
1L:1R	46.33	19.15	43.6	22.63	5.84	21.03	25.14	7.25	23.43
1L:1R, Auxl	64.04	28.32	60.55	33.23	10.01	30.3	36.50	11.95	33.51
1L:AR, AL:1R	43.99	20.4	42	21.16	5.48	19.71	23.58	7.06	22.07
1L:AR, AL:1R, Auxl	62.57	32.02	58.98	30.61	10.69	28.77	34.00	12.95	31.98
S-Tree, 1L:1R, Auxl	68.95	36.34	65.49	38.95	13.9	35.54	42.13	16.28	38.72
S-Tree+	70.38	38.79	66.4	40	14.75	36.34	43.22	17.30	39.53

OpenIE triples vs. Dependency Subtrees. We can first observe 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 OpenIE triples by +9, +7, +9 rouge-1,-2,-L respectively for extractive data, and +3, +1 and +3 points for 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 that is semantically correct with respect to the input sentence. Given that [3] achieved good scores on the Gigaword data and that S-trees outperform OpenIE triples, this suggests that S-Tree subtrees are potentially good alternatives to OpenIE triples.

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 summaries is much larger for extractive than abstractive data. This is in line with previous works [5, 6], which show that extractive compression 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 modeling of the paraphrases and variations occurring in abstractive compressions.

[3] `https://stanfordnlp.github.io/CoreNLP/openie.html`

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)

Auxiliary subtrees. We observe a large, significant improvement from (1L:1R) to (1L:1R, Auxl) and similarly, between (1L:AR, AL:1R) and (1L:AR, AL:1R, Auxl). In fact, this increase shows up systematically in all our experiments. This shows the importance of subtrees below the subject level, and that dependents and modifiers of these nodes often contain key information that is preserved in the compressed sentence.

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 respectively +10, +7, +9 rouge-1,-2-L points 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.

Qualitative analysis. Table 3 shows examples of multiple subtrees retrieved from the input document as well as the summaries generated from them. We can see that normal sequence-to-sequence with attention and copy mechanism struggles to identify important information and produces loops and repetitions in the end. On the other hand, thanks to the dependency structure, the summaries generated from subtrees contain short and coherent sentences.

7 Conclusion

We have proposed a flexible select-and-paraphrase summarization model that decouples the syntactic analysis process from the generation component, hence enabling plugging-in and out various syntactic parsing modules. We have demonstrated this flexibility by seamlessly exploiting both a full-blown dependency parser and the shallow OpenIE triples extractor. The dependency parser giving better results, we have further proposed multiple heuristics to extract from the syntactic tree the most informative subtrees for the task of summarization and analyzed experimentally their potential. Compared to the state-of-the-art end-to-end deep learning systems, the proposed approach has another advantage, as it may more easily explain its generated summaries by presenting to the user the actual subtrees that have lead to the output sentence. Although this approach can theoretically handle both extractive and abstractive summarization, we show that it is particularly effective on extractive types of summaries, and that more work is still required to improve the generator component of this architecture.

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