Coreference Resolution in Full Text Articles with BERT and Syntax-based Mention Filtering

Hai-Long Trieu[1], **Khoa N. A. Duong**[1], **Nhung T. H. Nguyen**[2], **Makoto Miwa**[1,3], **Hiroya Takamura**[1], **Sophia Ananiadou**[2]
[1]Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
[2]National Centre for Text Mining, University of Manchester, United Kingdom
[3]Toyota Technological Institute, Japan
{long.trieu, khoa.duong, takamura.hiroya}@aist.go.jp,
makoto-miwa@toyota-ti.ac.jp,
{nhung.nguyen, Sophia.Ananiadou}@manchester.ac.uk

Abstract

This paper describes our system developed for the coreference resolution task of the CRAFT Shared Tasks 2019. The CRAFT corpus is more challenging than other existing corpora because it contains full text articles. We have employed an existing span-based state-of-the-art neural coreference resolution system as a baseline system. We enhance the system with two different techniques to capture long-distance coreferent pairs. Firstly, we filter noisy mentions based on parse trees with increasing the number of antecedent candidates. Secondly, instead of relying on the LSTMs, we integrate the highly expressive language model–BERT into our model. Experimental results show that our proposed systems significantly outperform the baseline. The best performing system obtained F-scores of 44%, 48%, 39%, 49%, 40%, and 57% on the test set with B^3, BLANC, CEAFE, CEAFM, LEA, and MUC metrics, respectively. Additionally, the proposed model is able to detect coreferent pairs in long distances, even with a distance of more than 200 sentences.

1 Introduction

Coreference resolution is important not only in general domains but also in the biomedical domain. The Colorado Richly Annotated Full Text (CRAFT) corpus (Cohen et al., 2017) was constructed with an aim of boosting the performance of the task in the biomedical literature. Unlike other corpora, CRAFT is comprised of full text articles or full papers, its coreferent chains are arbitrarily long; the mean length of coreferent chains is 4 while the longest chain is 186, which makes the resolution even more difficult than usual. The corpus has been fully released in the CRAFT Shared Task 2019. In this paper, we present our approach to address the coreference resolution task in this challenging corpus.

We employ the state-of-the-art end-to-end coreference system (Lee et al., 2017) as our baseline. The system generates all continuous sequences of words (or spans) in each sentence as mention candidates, which means the number of candidates increases linearly to the number of sentences. Such candidates may contain a large number of noisy spans, which are spans in a sentence that do not fit any noun phrases according to the corresponding parse tree. Such noisy spans are often wasteful when being included in the list of candidates for the coreference resolution step. Especially for the CRAFT corpus, of which the average number of sentences is more than 300, the number of noisy spans would be many and needs to be reduced. Also, our observations on the CRAFT corpus show that in many cases, a mention and its antecedent are far away, e.g., a mention can occur in the result section of a paper while its antecedent is in the abstract section.

To address these problems, we enhance the baseline system in two ways; we propose to filter noisy spans by using syntactic information and increase the number of antecedent candidates to capture such long-distance coreferent pairs. We further boost the system by replacing the underlying Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) layer with the Bidirectional Encoder Representations from Transformer (BERT) model (Devlin et al., 2019)—a contextualized language model that can efficiently capture context in a wide range of NLP tasks.

We have evaluated our system on six common metrics for coreference resolution including B^3, BLANC, CEAFE, CEAFM, LEA, and MUC using the official evaluation script provided by the shared task organizers. By increasing the num-

Proceedings of the 5th Workshop on BioNLP Open Shared Tasks, pages 196–205
Hong Kong, China, November 4, 2019.

ber of antecedents and filtering noisy ones, we could boost the recall of mention detection, hence improving the performance of coreference resolution. When incorporating BERT into the system, we could attain better scores in both mention detection and coreference resolution at every metrics.

Our contributions are as follows.

- We proposed a new method to filter noisy spans, which is a weakness of the baseline system (Lee et al., 2017). Our filtering method based on syntactic trees reduced up to 90% noisy spans but still kept 93% of correct mentions on the development set. The method helps our model more computationally efficient than the baseline one, hence allowing us to increase the number of antecedent candidates to capture long-distance coreferent pairs.
- We successfully integrated the BERT model to replace the LSTM layers for coreference resolution task and obtained significant improvement.
- Although we only experimented our model with the CRAFT corpus, our proposed method is general enough to be applied to other corpora with long documents.

2 Methods

2.1 LSTM-based Baseline Model

Our model is based on the span-based end-to-end model (Lee et al., 2017). The model employs an exhaustive method to create any continuous sequences of words (spans) in each sentence. The representation of a span from the k-th word to the l-th word in a sentence is calculated by concatenating the information of the first word, last word, head word, and the span width feature as follows:

$$\boldsymbol{m}_{k,l} = [h_k, h_l, \hat{w}_{k..l}, \phi(k,l)]\,, \tag{1}$$

where h_k and h_l are embeddings of the first and last words calculated by a bidirectional LSTM; $\hat{w}_{k..l}$ is the weighted sum of the word vectors; and $\phi(k,l)$ encodes the size of this span.

Mention scores are calculated using a feed-forward neural network given the span representation.

$$\mathrm{s}_m(k,l) = w_m \cdot \mathrm{FFNN}_m(\boldsymbol{m}_{k,l}), \tag{2}$$

where w_m is a learnable weight vector; and FFNN denotes a feed-forward neural network.

Since the span-based model generates a large number of spans, a simple technique is used to rank and filter spans based on a λ ratio multiplied by the document size and choose the k best candidates.

To find an antecedent for each mention, we calculate the antecedent score as follows:

$$\begin{aligned}\mathrm{s}_a(\boldsymbol{m}_{k,l}, \boldsymbol{m}_{u,v}) = w_a \cdot \mathrm{FFNN}_a([\boldsymbol{m}_{k,l}, \boldsymbol{m}_{u,v},\\ \boldsymbol{m}_{k,l} \circ \boldsymbol{m}_{u,v}, \phi((k,l),(u,v))]),\end{aligned} \tag{3}$$

where w_a is a learnable weight vector; $\circ$ denotes an element-wise multiplication and $\phi((k,l),(u,v))$ represents the feature vector between the two mentions.

2.2 Coreference Resolution with BERT

Recently, BERT (Devlin et al., 2019) shows significant improvement on various tasks in comparison with other deep learning models including LSTMs. This highly expressive language model is able to capture contextual information effectively. We, therefore, aim at investigating whether this architecture can work effectively on coreference resolution in comparison with the previous LSTM-based models.

In the BERT model, contextual representations are assigned to sub-words in each word. We use the representation of the last subword in a word as the representation of the word and calculated the span representation using Equation 1. Since the pre-trained BERT model just supports sentences up to 512 sub-words, we utilize a sliding window technique with a window size of 512 and stride of 256 for longer sentences and then retrieve subword embeddings from windows so that each subword has maximum left and right context. We adapted the mention score and antecedent score functions as Equations 2 and 3.

2.3 Learning Parse Trees to Filter Mentions

A weakness of the span-based baseline model is that the greedy method generates a large number of noisy, mostly meaningless, spans. Although Lee et al. (2017) proposed to select k-best candidates but this strategy is problematic when working on long documents, in which a mention is probably far away from its true antecedents while there are a large number of noisy candidates between them.

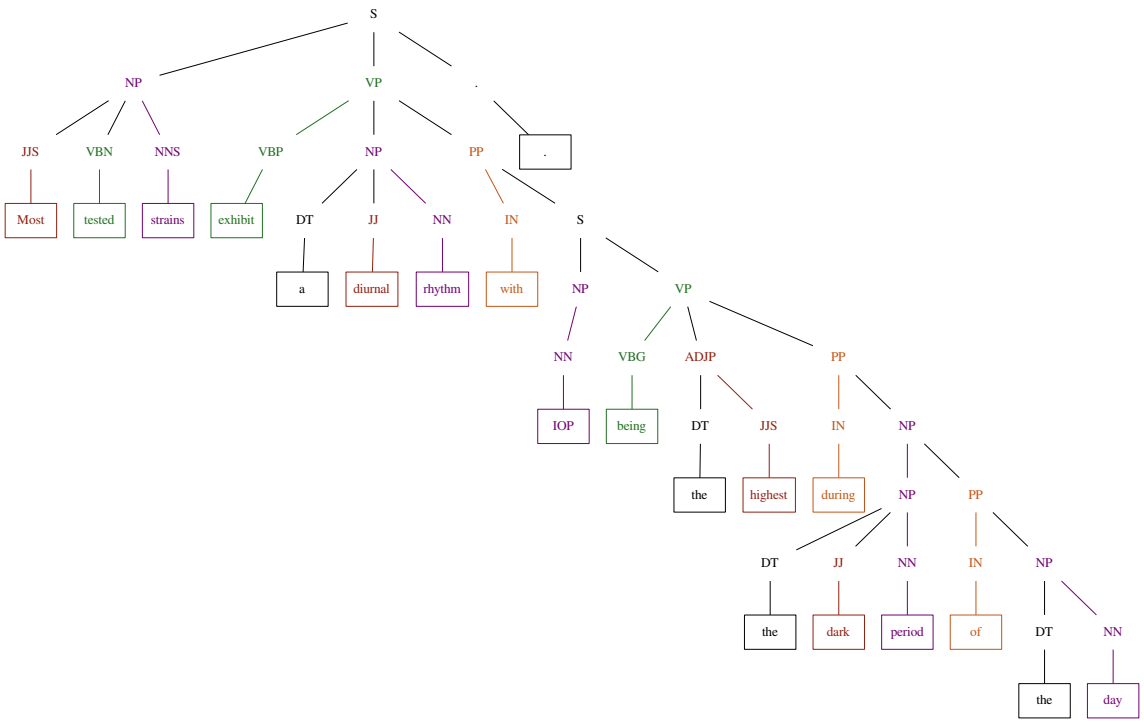

Figure 1: Three patterns corresponding to three gold mentions are extracted from the parse tree: "*a diurnal rhythm with IOP*" (pattern: **(NP,IN,NP)**), "*the dark period of the day*" (pattern: **(NP)**), "*the day*" (pattern: **(NP)**)

In order to overcome this issue, we propose to filter noisy spans based on their syntactic information. We observe that in the task like coreference resolution, mentions usually follow syntactic structures such as noun phrases. We therefore learn a syntactic parsing model to parse sentences and then extract patterns of gold mentions based on the resulting parse trees.

The end-to-end parsing model is trained jointly by the two following steps.

- **Part-of-speech (POS) classifier**: given raw sentences from the training set, words are split into sub-words with corresponding vectors from BERT embeddings. The last sub-word embedding of each word is used as the word embedding and passed through a linear layer to predict POS tags. The gold label POS tags are obtained from the CRAFT training set. Predicted POS tags and the raw texts will be used as the input for the parsing model.

- **Parser**: our model is based on the constituency parsing model (Kitaev and Klein, 2018), in which parse trees were built based on a self-attentive encoder and achieved state-of-the-art performance on the Penn Treebank. Unlike their model, we replaced the self-attentive encoder by BERT.

Figure 1 presents an example of using a parse tree to extract patterns of gold mentions. In this example, three patterns corresponding to three gold mentions are extracted: (NP, IN, NP), (NP), and (NP).

In the coreference resolution model, generated spans that match with the learned patterns are fed into the span representation layer to create span embeddings, while unmatched spans are ignored.

3 Experimental Settings

3.1 Dataset

The organizer provided two subsets of the CRAFT corpus (Cohen et al., 2017): one for training and one for testing systems. To estimate our model before submitting testing results, we further divided the original training set into two subsets, namely training and development sets. Table 1 shows the statistics numbers of these three subsets.

3.2 Compared Models

In order to show the effect of our proposed methods, we compare the following models.

	Train	Dev	Test
No. documents	60	7	30
No. sentences	19,575	2,156	9,099
Avg. sentences/doc	326.25	308.0	303.3
No. mentions	69,413	8,342	33,749
No. discon. men.	4,041	444	1,845
No. coreference.	14,679	1,623	7,185

Table 1: Characteristics of the CRAFT corpus. (*discon. men.*: discontinuous mentions)

- **LSTM**: this is the baseline model (Lee et al., 2017) based on LSTM. This model obtained the state-of-the-art performance on general domain. In this setting, all generated spans are used for calculating mention scores. The number of antecedent candidates was 250 and the $\lambda = 0.25$.
- **LSTM_filter**: this is the same as the LSTM baseline model, but we applied the filtering method and increased the antecedent number to 600 instead of 250.
- **BERT**: we employed the pre-trained SciBERT model (Beltagy et al., 2019) instead of using the LSTM as the baseline model. The number of antecedent candidates was 600.
- **BERT_filter**: we used the same settings as BERT but we combined it with the filtering method.
- **E2E_MetaMap** (Trieu et al., 2018): this model is based on the baseline model (Lee et al., 2017) but it particularly incorporated semantic type features extracted from the MetaMapLite (Demner-Fushman et al., 2017) to address biomedical documents. The maximum antecedent was 250.

The E2E_MetaMap implementation is based on the Tensorflow repository.[1] Meanwhile, the LSTM, LSTM_filter, and BERT_filter are based on the PyTorch repository.[2]

For the BERT model, we employed the PyTorch Pretrained BERT repository.[3] We trained the model with the Adam optimizer (Kingma and Ba, 2015). We included gradient clipping and dropout.

[1] https://github.com/kentonl/e2e-coref/tree/1f37582e68

[2] https://github.com/allenai/allennlp/tree/master/allennlp/models/coreference_resolution

[3] https://github.com/huggingface/pytorch-pretrained-BERT/tree/34cf67fd6c

Syntactic Patterns	Frequency	Ratio (%)
NP	40,639	58.55
NN	10,746	15.48
NML	3,462	4.99
PRP$	1,012	1.46
NN, NN	1,012	1.46
NP, NN	678	0.98
LS	647	0.93

Table 2: The most frequent patterns of mentions in the training set. Please check Appendix A for the definition of relevant Penn Treebank labels.

4 Results and Discussion

We firstly present the results of extracting patterns to filter mentions. We then report and discuss the performance of our models on the official test set. In order to deeply investigate the effect of the proposed method, we describe the intensive results of ablation tests on the development set. We finally conduct analysis to see how each model works on each group of sentence-level distance between mentions and antecedents.

4.1 Patterns of Gold Mentions

Table 2 reports some patterns[4] with the highest frequencies in the training set. In total, we extracted 1,561 unique patterns. To avoid low quality filtering, we kept patterns with a minimum frequency threshold of 5. The threshold was chosen from our experiments so that we could filter a large number of noisy spans but still kept a high recall on the development set. Specifically, this filtering method helps to reduce up to 90% noisy spans but still kept 93% of correct mentions on the development set.

4.2 Evaluation on the Test Set

The results on the official test set are presented in Table 3. In summary, our BERT_filter obtained the best performance on both mention and coreference detection in all metrics.

Mention detection For mention detection, most models obtained approximately the same precision of more than 70%. However, the recall

[4] The tag set in our patterns follows Penn Treebank POS tags.

Metric	Model	Mention			Coreference		
		P	R	F	P	R	F
B^3	LSTM	0.7565	0.3578	0.4858	0.6177	0.1583	0.2520
	LSTM_filter	0.7292	0.4187	0.5320	0.5764	0.2524	0.3511
	BERT	0.7416	0.5603	0.6383	0.5151	0.3544	0.4199
	BERT_filter	0.7314	0.5778	**0.6456**	0.5166	0.3838	**0.4404**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.5247	0.2791	0.3644
BLANC	LSTM	0.7565	0.3578	0.4858	0.6434	0.2153	0.3227
	LSTM_filter	0.7292	0.4187	0.5320	0.6471	0.3903	0.4869
	BERT	0.7416	0.5603	0.6383	0.5376	0.4350	0.4809
	BERT_filter	0.7314	0.5778	**0.6456**	0.5056	0.4731	**0.4888**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.5297	0.4140	0.4648
CEAFE	LSTM	0.7565	0.3578	0.4858	0.3590	0.2076	0.2631
	LSTM_filter	0.7292	0.4187	0.5320	0.4100	0.2408	0.3034
	BERT	0.7416	0.5603	0.6383	0.4366	0.3305	0.3762
	BERT_filter	0.7314	0.5778	**0.6456**	0.4544	0.3537	**0.3978**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.3545	0.3101	0.3308
CEAFM	LSTM	0.7565	0.3578	0.4858	0.5141	0.2431	0.3301
	LSTM_filter	0.7292	0.4187	0.5320	0.5847	0.3357	0.4265
	BERT	0.7416	0.5603	0.6383	0.5432	0.4104	0.4676
	BERT_filter	0.7314	0.5778	**0.6456**	0.5551	0.4385	**0.4900**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.4662	0.3662	0.4102
LEA	LSTM	0.7565	0.3578	0.4858	0.5733	0.1331	0.2161
	LSTM_filter	0.7292	0.4187	0.5320	0.5415	0.2265	0.3194
	BERT	0.7416	0.5603	0.6383	0.4692	0.3135	0.3759
	BERT_filter	0.7314	0.5778	**0.6456**	0.4753	0.3454	**0.4000**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.4864	0.2433	0.3244
MUC	LSTM	0.7565	0.3578	0.4858	0.6765	0.3007	0.4164
	LSTM_filter	0.7292	0.4187	0.5320	0.6656	0.3798	0.4837
	BERT	0.7416	0.5603	0.6383	0.6412	0.4842	0.5517
	BERT_filter	0.7314	0.5778	**0.6456**	0.6445	0.5111	**0.5701**
	E2E_MetaMap	0.6713	0.5272	0.5906	0.5995	0.4564	0.5182

Table 3: Results on the test set. The three official submissions of our team were BERT, BERT_filter and E2E_MetaMap. The non-coreference scores of BLANC are reported in Appendix B.

of the BERT_filter is much higher than those of the LSTM and LSTM_filter (57% vs. 35% and 41%, respectively). Consequently, the F-score of the BERT_filter is 16% and 11% points higher than the LSTM and LSTM_filter, respectively. The E2E_MetaMap is 5% points lower than the BERT_filter in F-score.

Coreference detection By obtaining the highest recall in mention detection, the BERT_filter could achieve the highest scores in coreference detection in all metrics. Using the mention filtering improved the baseline LSTM from 4-16% points in F-score varied by metrics. When replacing LSTM by BERT and combining with mention filtering, we obtained significant improvements: +19% points of B^3 and LEA; +16% points of MUC, BLANC and CEAFM; and +13% points of CEAFE in F-score.

The E2E_MetaMap performance is higher than the LSTM and LSTM_filter, but lower than the BERT_filter. As aforementioned, the LSTM model is based on the PyTorch implementation while the E2E_MetaMap is based on the Tensorflow repository. Therefore it is difficult to verify whether performance difference comes from using MetaMap features or from the implementation. Due to time constraint, we have not conducted experiments to clarify the reasons yet. We will leave this as our future work.

Metric	Model	Mention			Coreference		
		P	R	F	P	R	F
B^3	LSTM	0.7716	0.3782	0.5076	0.6387	0.1763	0.2764
	LSTM_filter	0.7193	0.4396	0.5457	0.5725	0.2572	0.3550
	BERT_250	0.7288	0.5675	0.6381	0.5258	0.32	0.3979
	BERT	0.7094	0.5742	0.6347	0.4807	0.3596	0.4115
	BERT_filter	0.7066	0.6014	**0.6498**	0.5021	0.3855	**0.4361**
BLANC	LSTM	0.7716	0.3782	0.5076	0.5665	0.1487	0.2356
	LSTM_filter	0.7193	0.4396	0.5457	0.5503	0.2235	0.3179
	BERT_250	0.7288	0.5675	0.6381	0.5129	0.3256	0.3983
	BERT	0.7094	0.5742	0.6347	0.4691	0.3168	0.3782
	BERT_filter	0.7066	0.6014	**0.6498**	0.5141	0.3757	**0.4341**
CEAFE	LSTM	0.7716	0.3782	0.5076	0.3659	0.2536	0.2996
	LSTM_filter	0.7193	0.4396	0.5457	0.4123	0.3252	0.3636
	BERT_250	0.7288	0.5675	0.6381	0.3888	0.3672	0.3777
	BERT	0.7094	0.5742	0.6347	0.4115	0.3674	0.3882
	BERT_filter	0.7066	0.6014	**0.6498**	0.4176	0.3993	**0.4083**
CEAFM	LSTM	0.7716	0.3782	0.5076	0.5285	0.2591	0.3477
	LSTM_filter	0.7193	0.4396	0.5457	0.5757	0.3518	0.4368
	BERT_250	0.7288	0.5675	0.6381	0.5073	0.3952	0.4443
	BERT	0.7094	0.5742	0.6347	0.5143	0.4163	0.4602
	BERT_filter	0.7066	0.6014	**0.6498**	0.5308	0.4518	**0.4881**
LEA	LSTM	0.7716	0.3782	0.5076	0.5974	0.1507	0.2407
	LSTM_filter	0.7193	0.4396	0.5457	0.5370	0.2276	0.3197
	BERT_250	0.7288	0.5675	0.6381	0.4811	0.2805	0.3544
	BERT	0.7094	0.5742	0.6347	0.4383	0.3196	0.3696
	BERT_filter	0.7066	0.6014	**0.6498**	0.4619	0.3464	**0.3959**
MUC	LSTM	0.7716	0.3782	0.5076	0.7065	0.3117	0.4325
	LSTM_filter	0.7193	0.4396	0.5457	0.6658	0.3783	0.4825
	BERT_250	0.7288	0.5675	0.6381	0.6418	0.4743	0.5455
	BERT	0.7094	0.5742	0.6347	0.6144	0.4850	0.5421
	BERT_filter	0.7066	0.6014	**0.6498**	0.6271	0.5179	**0.5673**

Table 4: Results on the development set.

4.3 Ablation Tests

We conducted experiments on the development set to show the effect of using mention filtering and BERT. In order to directly compare between BERT and LSTM, we also conducted an experiment with BERT and set a value of 250 to the number of antecedent candidates. We named it as BERT_250. Meanwhile, LSTM, LSTM_filter, BERT, BERT_filter have the same settings as described in Section 3.2. All of the results are reported in Table 4.

Mention Filtering When we used mention filtering, the mention detection precision dropped 6% points in the case of LSTM, but in the case of BERT it was almost the same. However, the filtering helped to improve recall in both cases, which is important to the coreference detection step. As a result, in the coreference resolution step, mention filtering improved 2-8% points of F-score in all metrics.

Using BERT Using BERT could significantly boost the performance of the baselines in both mention detection and coreference resolution. For mention detection, BERT produced almost the same precision with the LSTM but much higher recall (+17% points), which led to an increase of 10% points in F-score. For coreference detection, BERT-based models outperformed the LSTM-based ones from 4-14% points of F-scores in all metrics.

In summary, when combining both techniques

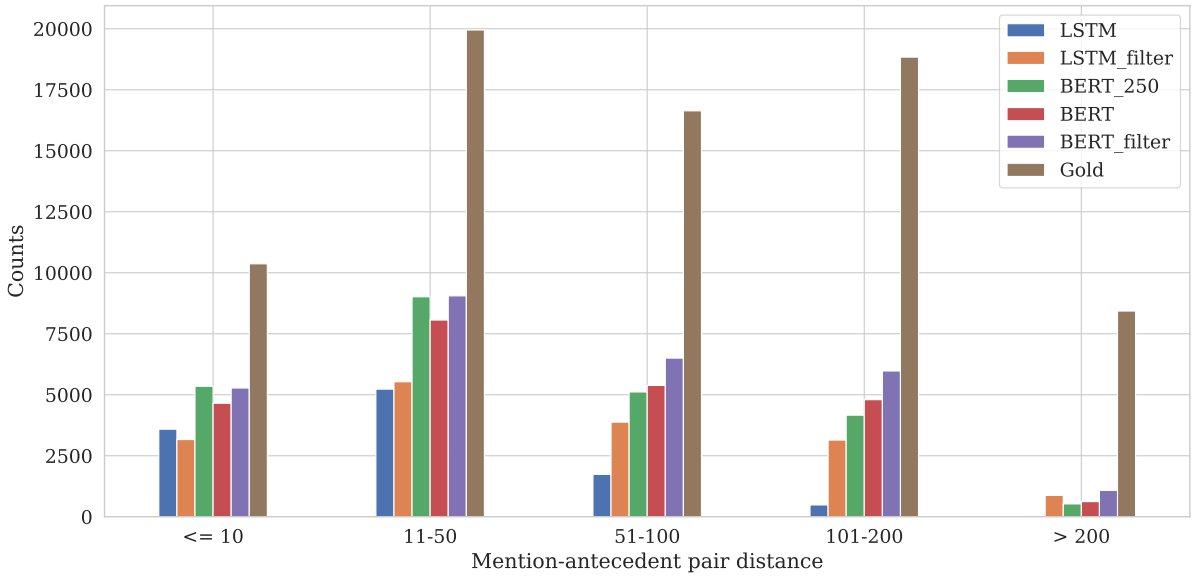

Figure 2: Numbers of correct predictions (true positives) grouped by sentence-level distance between a mention and its antecedent in the development set. The detail results are reported in Appendix C.

(BERT_filter vs. LSTM), we could make a significant increase of more than 14% points in F-score for mention detection and from 11% to 20% points in F-score in all metrics for coreference resolution on the development set.

4.4 Analysis

To investigate the effect of distance between a mention and its antecedent(s) on each model, we calculated the number of true positive coreference predictions in the development set and grouped them by sentence-level distance. Specifically, we divided the number of true positive predictions into five groups: $\leq$10, 11-50, 51-100, 101-200, and >200. The first two groups can be considered as short-distance coreference, e.g., abstract papers like the BioNLP dataset (Nguyen et al., 2011) with an average of nine sentences per document. Meanwhile, the other three groups can be considered as long-distance coreference like full papers in the CRAFT corpus.

Distribution of coreferent pairs in gold data As illustrated in Figure 2, only about 40.85% of the gold pairs are in the groups of short distance while the other 59.15% of them are in the groups of long distance. This means that if a model cannot deal with long distance coreference, pairs of mentions and antecedents in this region cannot be discovered. Among those long distance pairs, 47.8% are in between 51-200 sentences while the number of pairs whose distance is more than 200 is about 11.35%.

The effect of mention filtering The results in Figure 2 revealed that by using the filtering methods, we could effectively address long-distance coreferent pairs. It can be seen from the figure that the baseline model was good enough when working on short distance pairs, and the filtering may slightly harm the performance. However, for longer distances, the filtering contributed to increases of 5.46%, 55.26%, 84.60% and 100% for the groups of 11-50, 51-100, 101-200, and >200, respectively, in comparison with the baseline.

The effect of BERT Without using the filtering method, BERT itself could capture a fairly large number of long-distance pairs, which was even better than the LSTM_filter model.

Long-distance coreference When summing up the results of long-distance groups, i.e., three groups of 51-100, 101-200, and >200, we found that the LSTM_filter and BERT_filter models could predict 71.89% and 83.63% higher number of long-distance pairs than the LSTM one, which indicates the effect of our model in using the filtering method and BERT for long documents. Additionally, for the most tough case of detecting pairs in the distance of more than 200 sentences, our LSTM_filter model predicted 874 correct pairs (about 1.18% of the gold pairs), and the BERT_filter model predicted 1,081 correct pairs (about 1.46% of the gold pairs). Meanwhile, the LSTM model failed to detect pairs in this group.

Recall problem It is necessary to note that although our model improved the baseline LSTM and obtained promising results, the recall is still low in all groups of distance. For instance, the best performing model, i.e., BERT_filter could cover only 50.90%, 45.38%, 39.08%, 31.71%, and 12.82% of the gold pairs in the groups of <=10, 11-50, 51-100, 101-200, >200, respectively. This is an open issue that we will address in the future.

5 Conclusion

In this paper, we particularly address the challenge of coreference resolution in full text articles in the CRAFT Shared Task 2019. Specifically, we employ the span-based end-to-end model (Lee et al., 2017) and enhance the model by utilizing a syntax-based mention filtering method and BERT. To filter noisy mentions, we jointly train a parsing model with a POS classifier to obtain parse trees of sentences. We then generate syntactic patterns of gold mentions based on the resulting parse trees. Any mentions that satisfy the generated patterns will be fed into the coreference resolution model. We finally incorporate BERT into our model. Experimental results on the CRAFT corpus indicate that the proposed method is effective in capturing long-distance coreferences in long documents.

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work is also supported by PRISM (Public/Private R&D Investment Strategic Expansion PrograM).

References

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

K. Bretonnel Cohen, Arrick Lanfranchi, Miji Joo-young Choi, Michael Bada, William A. Baumgartner Jr., Natalya Panteleyeva, Karin Verspoor, Martha Palmer, and Lawrence E. Hunter. 2017. Coreference annotation and resolution in the colorado richly annotated full text (CRAFT) corpus of biomedical journal articles. *BMC Bioinformatics*, 18(1):372:1–372:14.

Dina Demner-Fushman, Willie J Rogers, and Alan R Aronson. 2017. MetaMap Lite: an evaluation of a new Java implementation of MetaMap. *Journal of the American Medical Informatics Association*, 24(4):841–844.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR2015)*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Ngan Nguyen, Jin-Dong Kim, and Jun'ichi Tsujii. 2011. Overview of BioNLP 2011 protein coreference shared task. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 74–82, Portland, Oregon, USA. Association for Computational Linguistics.

H-L. Trieu, N. T. H. Nguyen, M. Miwa, and S. Ananiadou. 2018. Investigating domain-specific information for neural coreference resolution on biomedical texts. In *Proceedings of the BioNLP 2018 workshop*, pages 183–188.

Colin Warner, Ann Bies, Christine Brisson, and Justin Mott. 2004. Addendum to the penn treebank ii style bracketing guidelines: Biomedical treebank annotation. *University of Pennsylvania*.

A Penn Treebank Labels

For the Penn Treebank labels in our syntactic patterns, we follow the BioMedical Treebank tagset definition (Warner et al., 2004). Please refer to Table 5 for the detail description.

B Non-Coreference Results

Unlike other metrics, the BLANC metric also contains non-coreference results. We report the results of the test set in Table 6.

C Results on (Mention-Antecedent) Pair Distance

We present the detail results of each model and the corresponding gold coreference grouped by the sentence-level distance of mention-antecedent pairs in Table 7. The results are calculated in five groups of distance: $<=$10, 11-50, 51-100, 101-200, $>$200.

Tag	Description
NP	noun phrase
NN	noun, singular or mass
NML	sub-NP nominal substrings
PRP$	possessive pronoun
LS	list item marker

Table 5: The definition of relevant Penn Treebank labels.

Metric	Model	Mention			Non-Coreference			
		P	R	F	P	R	F	BLANC-F
BLANC	LSTM	0.7565	0.3578	0.4858	0.5725	0.1314	0.2137	0.2682
	LSTM_filter	0.7292	0.4187	0.5320	0.5432	0.1796	0.2699	0.3784
	BERT	0.7416	0.5603	0.6383	0.5537	0.3172	0.4033	0.4421
	BERT_filter	0.7314	0.5778	0.6456	0.5383	0.3356	0.4135	0.4512
	E2E_MetaMap	0.6713	0.5272	0.5906	0.4488	0.2812	0.3457	0.4053

Table 6: Non-coreference results for the BLANC metric on the testing set.

Model	<=10			11-50			51-100		
	TP	G.R (%)	O.R (%)	TP	G.R (%)	O.R (%)	TP	G.R (%)	O.R (%)
LSTM	3,588	34.61	4.83	5,230	26.22	7.05	1,735	10.43	2.34
LSTM_filter	3,164	30.52	4.26	5,532	27.73	7.45	3,878	23.31	5.23
BERT_250	5,347	51.58	7.20	9,015	45.19	12.15	5,115	30.75	6.89
BERT	4,651	44.87	6.27	8,062	40.41	10.86	5,381	32.35	7.25
BERT_filter	5,276	50.90	7.11	9,053	45.38	12.20	6,502	39.08	8.76
Group Gold	10,366	100	13.97	19,949	100	26.88	16,636	100	22.41

Model	101-200			>200		
	TP	G.R (%)	O.R (%)	TP	G.R (%)	O.R (%)
LSTM	484	2.57	0.65	0	0.00	0.00
LSTM_filter	3,143	16.69	4.23	874	10.37	1.18
BERT_250	4,163	22.10	5.61	527	6.25	0.71
BERT	4,798	25.47	6.46	620	7.35	0.84
BERT_filter	5,974	31.71	8.05	1,081	12.82	1.46
Group Gold	18,837	100	25.38	8,431	100	11.36
Total Gold	74,219					

Table 7: Results of models on each distance group; **TP**: True Positive; **G.R**: Group Ratio = True Positive/Group Gold; **O.R**: Overall Ratio = True Positive/Total Gold