Antecedent Prediction Without a Pipeline

Sam Wiseman and **Alexander M. Rush** and **Stuart M. Shieber**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,srush,shieber}@seas.harvard.edu

Abstract

We consider several antecedent prediction models that use no pipelined features generated by upstream systems. Models trained in this way are interesting because they allow for side-stepping the intricacies of upstream models, and because we might expect them to generalize better to situations in which upstream features are unavailable or unreliable. Through quantitative and qualitative error analysis we identify what sorts of cases are particularly difficult for such models, and suggest some directions for further improvement.

1 Introduction

Most recent approaches to identity coreference resolution rely on a set of pipelined features generated by relatively accurate upstream systems. For instance, the CoNLL 2012 coreference datasets (Pradhan et al., 2012), which are based on the OntoNotes corpus (Hovy et al., 2006), make available both gold and predicted parse, part-of-speech, and named-entity information for each sentence in the corpus. While recent systems have managed to improve on the state of the art in coreference resolution by taking advantage of such information (Durrett and Klein, 2013; Wiseman et al., 2015; Björkelund and Kuhn, 2014; Fernandes et al., 2012; Martschat and Strube, 2015), we might be interested in systems that do not use pipelined features for several reasons: first, pipelined systems are known to accumulate errors throughout the stages of the pipeline. Second, unpipelined models do not need to contend with the intricacies of the various systems in the pipeline, which may have little impact on the target task. Finally, models that do not require pipelined features may be more applicable to regimes in which upstream features are unavailable or unreliable, such as those arising from predicting coreference in low-resource languages or in social media text. Indeed, to the extent that it is easier to obtain coreference annotations than it is to obtain (for instance) parse annotations in such regimes, an unpipelined strategy may be particularly practical.

Accordingly, in this paper we consider systems that attempt to move beyond OntoNotes by making coreference predictions without access to pipelined features, using only a document's words and sentence boundaries. In the hopes of shedding light on whether this is a viable strategy, we consider, as a case study, how well coreference systems without access to upstream features can perform on English. Given the amount of research that has gone into resolving English coreference resolution *with pipelined features*, by also considering the English "unpipelined" setting we can expect to get a rather accurate sense of how much we sacrifice by ignoring these features. Moreover, in addition to the benefits of unpipelined models noted above, the proposed line of research is congenial to the recent trend in NLP of using as few hand-engineered features as possible (as advocated, for instance, in Collobert et al. (2011)).

We report preliminary experiments on the subtask of antecedent prediction (defined in Wiseman et al. (2015) and reviewed below) on the CoNLL 2012 English dataset in this unpipelined setting. In particular, we will assume that we have automatically

Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016), co-located with NAACL 2016, pages 53–58,
San Diego, California, June 16, 2016.

extracted mentions from a document, but that no other pipelined information is available. We emphasize that this is a strong assumption (since pipelined features, such as parse trees, are often used to extract mentions), and so what follows should be interpreted as an attempt to obtain an upper bound on the performance possible in such a setting. We conclude by analyzing the errors made by the proposed unpipelined systems, and discussing how these systems might be made more competitive.

1.1 Problem Setting

As above, we will assume we are given a set of documents from which we are able to automatically extract mentions. We denote by $\mathcal{X}$ the set of these automatically extracted mentions. For a mention $x \in \mathcal{X}$, let $\mathcal{A}(x)$ denote the set of mentions appearing before x in the document, and let the set $\mathcal{C}(x) \subseteq \mathcal{A}(x)$ denote the mentions appearing before x that are coreferent with x. The problem of antecedent ranking involves trying to predict an antecedent $y \in \mathcal{C}(x)$ for only those x for which $\mathcal{C}(x) \neq \emptyset$, that is, for only those x that have coreferent antecedents. We will moreover require that in making these antecedent predictions no pipelined features are used. In particular, we will assume that "unpipelined" systems have access only to a document's mention-boundaries, to the sets $\mathcal{C}(x)$ for each $x \in \mathcal{X}$ (when training), to the words in each document, and to the document's sentence boundaries.

Whereas recent coreference systems typically make use of syntactic information, named-entity tags, word-lists containing type information (e.g., number, gender, animacy), and speaker information (Durrett and Klein, 2013; Björkelund and Kuhn, 2014; Lee et al., 2013), given the aforementioned restrictions, the only common coreference features that remain legal are word-based features and "distance" features. Distance features are typically defined in terms of the number of words, mentions, or sentences between a mention and a candidate antecedent (Durrett and Klein, 2013), and such features can presumably be defined accurately in many settings without the use of upstream systems.

2 Models

We will use a very simple mention-ranking style model for our antecedent prediction. Mention-ranking models make use of a scoring function $s(x, y)$ that scores the compatibility between a mention x and a candidate antecedent y, and they predict the antecedent to be $y^* = \arg\max_{y \in \mathcal{C}(x)} s(x, y)$. We will define s as

$$s(x, y) = \boldsymbol{u}^\mathsf{T} \tanh\left(\boldsymbol{W}\begin{bmatrix}\boldsymbol{\Phi}_c(x)\\ \boldsymbol{\Phi}_c(y)\\ \boldsymbol{\Phi}_d(x, y)\end{bmatrix} + \boldsymbol{b}\right),$$

where $\boldsymbol{\Phi}_c$ extracts relevant word-based features from a mention and its context, and $\boldsymbol{\Phi}_d$ extracts distance based features between x and y. Thus, the scoring function s is defined by applying a standard multi-layer perceptron (MLP) to the (vertically) concatenated outputs of the functions $\boldsymbol{\Phi}_c$ and $\boldsymbol{\Phi}_d$. In particular, $\boldsymbol{W}$ represents the weight matrix of the MLP's first hidden layer, $\boldsymbol{b}$ the corresponding bias vector, and $\boldsymbol{u}$ the vector of weights projecting the first hidden layer into a scalar score. The exact dimensions of these weights will become clear in what follows.

In defining $\boldsymbol{\Phi}_c$ we will view a mention x spanning M words as a sequence of real vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$, with each $\boldsymbol{x}_m \in \mathbb{R}^D$ obtained by looking up the m'th word in x in an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{D \times |\mathcal{V}|}$, where $\mathcal{V}$ is our fixed vocabulary. Accordingly, let $\boldsymbol{X}_{1:M} \in \mathbb{R}^{D \times M}$ be the matrix formed by concatenating the embeddings of the words in a mention (in order). Analogously, let $\boldsymbol{X}_{-K:-1} \in \mathbb{R}^{D \times K}$ be the concatenation of the embedding-vectors corresponding to the K words preceding x on the left (padded where necessary), and $\boldsymbol{X}_{M+1:M+K}$ the concatenation of the embedding-vectors corresponding to the K words following x on the right (padded where necessary).

For simplicity, we will require $\boldsymbol{\Phi}_c$ to take the following form:

$$\boldsymbol{\Phi}_c(x) = \begin{bmatrix}\boldsymbol{h}(\boldsymbol{X}_{1:M})\\ \boldsymbol{h}(\boldsymbol{X}_{-K:-1})\\ \boldsymbol{h}(\boldsymbol{X}_{M+1:M+k})\end{bmatrix},$$

where $\boldsymbol{h}(\boldsymbol{X}_{i:j})$ is some function of the matrix $\boldsymbol{X}_{i:j}$. That is, $\boldsymbol{\Phi}_c(x)$ simply concatenates a representation

of the words of x with representations (respectively) of the K words preceding and following x.

For example, consider the following passage from the development portion of the CoNLL 2012 English development data, from which the final example in Table 1 is taken, and in which we have highlighted a particular mention we might like to predict an antecedent for:

> Suddenly we realized water came into the engine room and it was rising and they started to pump, of course, and they pumped and pumped and **the water** came more and more and more. (bn/cnn/cnn_0410)

If we are interested in predicting coreferent antecedents for "the water," which we will denote by x, then we will have $M = 2$, and $\boldsymbol{X}_{1:2}$ will be a matrix in $\mathbb{R}^{D\times 2}$ with its first column equal to the embedding (in $\boldsymbol{E}$) for "the," and its second column equal to the embedding for "water." Since in predicting x we will likely also want to take into account some of its surrounding context, we will also form matrices corresponding to the K words to the left and to the right (respectively) of x. Thus, if we set $K = 1$, we will form $\boldsymbol{X}_{-1:-1}$ as the matrix in $\mathbb{R}^{D\times 1}$, which consists of the embedding for "and," and we would define $\boldsymbol{X}_{M+1:M+1}$ analogously. Given the aforementioned $\boldsymbol{X}$ matrices, we define $\boldsymbol{\Phi}_c$ by vertically concatenating the output of applying a function $\boldsymbol{h}$ to each of these matrices.

We now consider three approaches to defining $\boldsymbol{h}(\boldsymbol{X}_{1:M})$, in increasing order of complexity:

Max-Over-Time Model: Define $\boldsymbol{h}(\boldsymbol{X}_{1:M})$ to be in $\mathbb{R}^D$, with $\boldsymbol{h}(\boldsymbol{X}_{1:M})_d = \max_{1\le j\le M} (\boldsymbol{X}_{1:M})_{dj}$, for each $d = 1, \ldots, D$.[1]

Convolutional Model: We follow Kim (2014) in generating F feature maps of $M - h + 1$ features by applying a (non-linear) filter to each h-length window of $\boldsymbol{X}_{1:M}$, and then max-pooling over time. Thus, $\boldsymbol{h}(\boldsymbol{X}_{1:M}) \in \mathbb{R}^F$.

LSTM Model: We define $\boldsymbol{h}(\boldsymbol{X}_{1:M})$ to be in $\mathbb{R}^H$, where $\boldsymbol{h}(\boldsymbol{X}_{1:M})$ is the M'th hidden state of an LSTM (Hochreiter and Schmidhuber, 1997) run over the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ in $\boldsymbol{X}_{1:M}$.

[1] We found the max-pooling described here to be more effective than mean-pooling.

To define $\boldsymbol{\Phi}_d$ we first define indicator features (represented as one-hot vectors), which (respectively) bucket the number of mentions and the number of sentences between a mention and a candidate antecedent into 11 discrete buckets, following Durrett and Klein (2013). We therefore have 22 distance indicator features in total, and they are used to index into an embedding matrix $\boldsymbol{A} \in \mathbb{R}^{D_d\times 22}$. Accordingly, $\boldsymbol{\Phi}_d(x, y) \in \mathbb{R}^{D_d}$ represents the sum of the (two) distance embeddings obtained from $\boldsymbol{A}$ in this way. This approach resembles that of Sukhbaatar et al. (2015).

3 Experiments

3.1 Methods

We conduct antecedent-ranking experiments on the development portion of the CoNLL 2012 English corpus. Mentions were extracted using the Berkeley Coreference System (Durrett and Klein, 2013). We set $K = 4$ in forming word-windows, and we trained by optimizing the margin ranking-loss defined in Wiseman et al. (2015) using mini-batch Adagrad (Duchi et al., 2011).

For the convolutional model, we used windows of size 1, 2, and 3, and 40 filters for each. We set D_d, the dimensionality of the distance feature embeddings which constitute the columns of $\boldsymbol{A}$, to 20. We used the `element-rnn` RNN package (Léonard et al., 2015) to implement the LSTM, and we set the LSTM's hidden-layer size to 200. All models used 300 hidden units in the final layer (represented by $\boldsymbol{W}$), and we used Dropout for regularization. All hyperparameters including window size were tuned on the development set.

For all models we initialized $\boldsymbol{E}$, the word embedding matrix, with word vectors obtained from `word2vec` (Mikolov et al., 2013), and so $\boldsymbol{E} \in \mathbb{R}^{300\times|\mathcal{V}|}$, where $\mathcal{V}$ is the vocabulary consisting of words in the training or development sets (plus an unknown word token). $\boldsymbol{E}$ was updated during training. For the Max-Over-Time Model we found it beneficial to untie the embedding matrices used to embed the words in the mention, before the mention, and after the mention, giving 3 separate embedding matrices. For the Convolutional and LSTM Models, performance was at least as good when using a single embedding matrix.

x	Correct Antecedent Prediction	Convolutional Antecedent Prediction
the Straits [Foundation]	the Straits [Foundation]	the Straits [Association]
those Jewish [sacrifices]	the [sacrifices]	the [people] of Israel
the [water]	[water]	their sinking fishing [boat]

Table 1: Example mentions x which the baseline MLP correctly predicts (middle column), but the Convolutional Model (right column) does not. Heads of each mention (unseen by the Convolutional Model) are in brackets.

Model	Acc.
Wiseman et al. (2015)	82.58
Max-Over-Time Model	70.92
Convolutional Model	72.65
LSTM Model	77.40

Table 2: Accuracy of models described in text (and baseline) on predicting antecedents on CoNLL Development set.

	Errors		
	HM	No HM	Pron.
Wiseman et al. (2015)	588	522	1146
Max-Over-Time Model	1513	608	1646
Convolutional Model	1358	607	1577
LSTM Model	1028	537	1362
Total Mentions	4677	973	7302

Table 3: Errors of models described in text on CoNLL 2012 development set. Mentions are partitioned column-wise as nominal or proper with (previous) head match in the document (HM), nominal or proper with no previous head match in the document (No HM), and pronominal.

3.2 Results

We are particularly interested in determining in what situations a word-and-distance model underperforms models with access to more sophisticated information. In Table 2 we compare the antecedent-prediction accuracy of the three models defined above with the antecedent ranking performance of the model described in Wiseman et al. (2015), which uses an MLP over pipelined coreference features. We will refer to this latter model as the "baseline MLP." We see that the word-and-distance models underperform, though the LSTM model comes within 5.2% of the baseline MLP. (It is also worth noting here that without the distance features Φ_d all models are significantly less accurate, with accuracies decreasing by over 15 percentage points).

4 Discussion

In Table 3 we examine, using an analysis similar to that in Durrett and Klein (2013), where the unpipelined models go wrong. There, we partition mentions column-wise into nominal or proper mentions that have a head-match with some previously occurring mention, nominal or proper mentions that do not, and pronominal mentions. (Note that whereas parse information must be used to detect heads, this is only used in our analysis, and none of the three models introduced here have access to this information).

Let us first consider the Convolutional Model, which underperforms the baseline in all categories, but does particularly badly in predicting antecedents for mentions for which a previous mention in the text has the same head.

Why is this? Further analysis shows that almost 84% of the HM examples that are correctly predicted by the baseline MLP but incorrectly predicted by the Convolutional Model involve the baseline MLP predicting an antecedent with an exact head-match to the current mention, and the Convolutional Model predicting a non-head-match antecedent. We show some representative examples in Table 1, where we bracket the head of each mention. As is evident from Table 1, the model is picking antecedents that are semantically reasonable, but which do not have a head match. The reason the Convolutional Model makes these errors is presumably that it is not able to tell what the head of each mention is (because it sees only the words in the mention, and the word-windows preceding and following). The baseline MLP, however, does have access to the heads of each mention, and so can learn that head-match is a discriminative feature.

As we move to the LSTM model, we find that errors decrease in all categories, though follow largely the same pattern. Indeed, over 78% of the LSTM

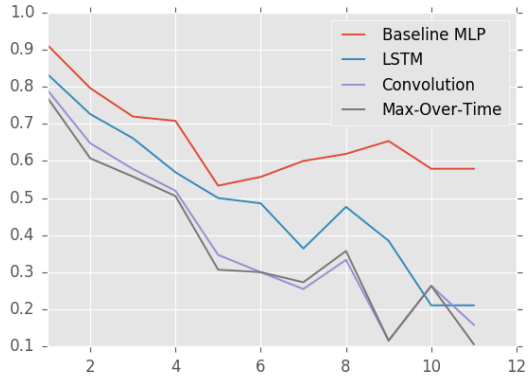

Figure 1: Percentage of antecedents in the CoNLL 2012 development set predicted correctly, by mention length.

model's errors in the HM category also involve predicting a non-head-match antecedent when the baseline MLP correctly predicts a head-match antecedent. Thus, it seems the LSTM model too could benefit from better head-finding. As additional evidence for this hypothesis, in Figure 1 we plot the percentage of correctly predicted antecedents in the CoNLL 2012 development set as the length of the current mention x increases. (Only mention-lengths occurring ≥ 10 times in the development set are reported). We see that the accuracy of both the Convolutional and LSTM models (as well as that of the Max-Over-Time model) generally decreases as the mention-length increases, though that of the baseline MLP model does not. Of course, it stands to reason that finding heads is more difficult in longer mentions, which may explain this trend.

When it comes to the other major category of errors in Table 3, namely, errors on pronominal mentions, it is more difficult to diagnose a single underlying cause of error. In particular, the unpipelined models' errors tend to involve either predicting antecedents that are inconsistent in terms of gender or number, or, interestingly, predicting non-pronominal antecedents when the baseline MLP predicts a pronominal antecedent. While it is certainly the case that the baseline MLP has access to gender information that the unpipelined models do not, it is not as clear why these unpipelined models learn to disprefer predicting pronominal antecedents for pronominal mentions, and this issue requires further investigation.

5 Conclusion

The results presented above suggest that a major factor holding word-and-distance-only models back from competing with models that have access to pipelined features is their inability to find mention-heads and, more generally, to take advantage of syntactic features. While the fact that such models would benefit from syntactic information is not surprising, the examples in Table 1 suggest that even coarse notions of head-finding may be sufficient to improve performance. Accordingly, one might imagine that alignment or attention models (such as that of Bahdanau et al. (2014)) that attempt to model coarse head-information would be useful in such cases.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference Resolution with Latent Antecedents and Non-local Features. *ACL, Baltimore, MD, USA, June*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett and Dan Klein. 2013. Easy Victories and Uphill Battles in Coreference Resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 41–48. Association for Computational Linguistics.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% Solution. In *Proceedings of the human language technology conference of the NAACL, Compan-*

ion Volume: Short Papers, pages 57–60. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1746–1751.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic Coreference Resolution based on Entity-centric, Precision-ranked Rules. *Computational Linguistics*, 39(4):885–916.

Nicholas Léonard, Yand Waghmare, Sagar ad Wang, and Jin-Hwa Kim. 2015. rnn: Recurrent Library for Torch. *arXiv preprint arXiv:1511.07889*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *TACL*, 3:405–418.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1416–1426.