Triad-based Neural Network for Coreference Resolution

Yuanliang Meng
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
ymeng@cs.uml.edu

Anna Rumshisky
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

Abstract

We propose a triad-based neural network system that generates affinity scores between entity mentions for coreference resolution. The system simultaneously accepts three mentions as input, taking mutual dependency and logical constraints of all three mentions into account, and thus makes more accurate predictions than the traditional pairwise approach. Depending on system choices, the affinity scores can be further used in clustering or mention ranking. Our experiments show that a standard hierarchical clustering using the scores produces state-of-art results with MUC and B^3 metrics on the English portion of CoNLL 2012 Shared Task. The model does not rely on many handcrafted features and is easy to train and use. The triads can also be easily extended to polyads of higher orders. To our knowledge, this is the first neural network system to model mutual dependency of more than two members at mention level.

1 Introduction

Entity coreference resolution aims to identify mentions that refer to the same entity. A mention is a piece of text, usually a noun, a pronoun, or a nominal phrase. Resolving coreference often requires understanding the full context, and sometimes also world knowledge not provided in the text. Generally speaking, three types of models have been used for coreference resolution: pairwise models, mention ranking models, and entity-mention models. The first two are more common in literature, and the third one is somewhat less studied.

Pairwise models a.k.a. mention pair models build a binary classifier over pairs of mentions (Soon et al., 2001; McCallum and Wellner, 2003). If all the pairs are classified correctly, then all coreferent mentions are identified. The mention ranking models do not rely on the full pairwise classification, but rather compare each mention to its possible antecedents in order to determine whether the mention might refer to an existing antecedent or starts a new coreference chain (Durrett and Klein, 2013; Wiseman et al., 2016; Clark and Manning, 2016). The entity-mention models try constructing representations of discourse entities, and associating different mentions with the entity representations (Luo et al., 2004).

However, none of these model types consider more than two mentions together at the low level. By low level here, we mean the processing of input mention features, as opposed to processing of constructed representations. Pairwise models and mention ranking models make low-level decisions on mention pairs only. Some further processing may be applied to reconcile global scope conflicts, but this process no longer relies directly on mention features.

This paper proposes a neural network model which works on triads of mentions directly. Each time, the system takes three mentions as input, and decisions on their coreference relations are made while taking into account all mutual dependencies. Inferences drawn from three mentions, if correctly modeled, should be more reliable than those from two mentions, simply because entities in a text tend to have multiple mutual dependencies. Firstly, coreference relation is transitive, and transitivity can be revealed only by 3 or more participants. Secondly, mutual dependencies are not just at the level of transitivity,

Proceedings of the 27th International Conference on Computational Linguistics, pages 35–43
Santa Fe, New Mexico, USA, August 20-26, 2018.

but can occur among lexical items, syntactic structures, or discourse information. Modeling dependency at these lower levels can therefore be helpful for coreference resolution. We believe it is also a closer approximation of humans' cognitive process. When we read text, we often look in two or more places (including not only mentions, but also their context) to decide what a pronoun might refers to. Therefore it is reasonable to account for it at an early stage of system design.

We show that the decisions made by the triad model are more accurate than those made by the dyad model. Such decisions can be further used in mention ranking, or simply followed up by clustering or graph partitioning as in the canonical mention pair models. The triad system can be easily extended to higher order polyads, if necessary. In this paper, we only consider triads, and dyads (pairs) are used for comparison. We use the English portion of CoNLL 2012 Shared Task dataset for training and evaluation. Our experiments show that a standard hierarchical clustering algorithm using the triad model output achieves state-of-art performance under several evaluation measures.[1]

2 Related Work

Before the neural network models became popular in coreference resolution tasks, graphical models had often been used to capture dependencies. McCallum and Wellner (2003) described a system which draws pairwise inferences but also accounts for transitivity constraints. Essentially, their model can be summarized as in equation

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_x} \exp\left(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij}) + \sum_{i,j,k,l'} \lambda_{l'} f_{l'}(y_{ij}, y_{jk}, y_{ik})\right) \quad (1)$$

The first term describes the potential function of a mention pair x_i, x_j as well as their label y_{ij}. For instance, $y_{ij} = 1$ if x_i and x_j belong to the same entity. The second term adds constraints on the labels to assure logical consistency. A particular assignment of values to x_i, x_j does not only affect the potential function involving these two nodes, but also other potential functions involving one of them. This makes the variables (mentions, in this case) dependent on each other. Exact algorithms to solve such problems are NP-hard, and some approximation techniques are often applied.

Our proposed model can be viewed as constructing potential functions over three variables x_i, x_j and x_k. However, we do not look to optimize the product of all the potential functions. Instead, we train a neural network model to assign labels to all edges within a triad locally. Note that the label y_{ij} for a given mention pair x_i, x_j may have different optimal values when different x_k's are used to construct a triad. The final assignment is determined by computing the average of y_{ij}'s. Moreover, our input features are mainly series of word embeddings and part of speech (POS) embeddings, encoding rich context information. The conventional algorithms used in graphical models cannot deal with such high dimensional features.

Graphical neural network (GNN) models have recently been used to process graphs (Duvenaud et al., 2015; Santoro et al., 2017; Kipf et al., 2018). For example, the graph convolutional networks (GCN) (Kipf and Welling, 2016; Defferrard et al., 2016) take graphs as input. Each node of the graph contains features, and a matrix represents their mutual relations. The features and the relation matrix are both used as input. Some filter layers, often shared by all nodes, process the features. The output of the filters and the relation matrix are further processed by other layers. The final output is new representations of nodes, which can be labels.

Our model shares some characteristics with GCNs. The triad input can be viewed as a basic graph: triangle, and each node is a mention. The features we used (word embeddings, POS embeddings, speaker identity, mention distance) are all associated with a node or a pair of nodes (edges). The three nodes share recurrent neural network layers. Because the output of such layers are used together, higher layers in the system have access to information from all the nodes. The output is a 3 dimensional binary vector, which can be considered a graphical representation too. However, our goal here is to find pairwise relations, and the triangle graphs are employed only to model (partial) mutual dependency among three mentions.

[1] Our source code is freely available here: `https://github.com/text-machine-lab/entity-coref`

In contrast, CGNs are capable of generating new complex representations for the nodes and they rely on the structure of the input graph, both of which are not applicable in our case.

3 System

The system consists of two major parts: the triad-based neural network model to compute mutual distances and a model to perform clustering. These two stages are not clearly divided, since defining mutual distances affects the clustering strategy.

3.1 Input Features

Our input is triads of entity mentions. The triads have mutual (joint) features and individual features as input. Speaker identity and mention distance are mutual features. The files in the CoNLL 2012 dataset are largely transcripts of broadcast news and conversations, which typically involve several speakers. We use a binary feature to indicate whether two mentions are from utterances of the same speaker (1) or not (0). Mention distance indicates how far apart two mentions are in the text. If they are next to each other, the distance is 1; if there is another mention in between, the distance is 2, and so on. We only count the mentions in between, regardless the number of the words or sentences in between.

Individual features are word tokens and POS tokens for each entity mention. The word tokens include the mentions themselves, as well as their 5 preceding tokens and 5 succeeding tokens. We also design two special tokens to mark the beginning and end of each mention. Similarly, the POS tokens include the POS tags of the mentions, as well as the POS tags of 5 preceding and 5 succeeding tokens. Two other special tags are used to mark the beginning and end of the mentions for POS tokens too.

Each word token is represented by a 300-dimensional vector. We use glove.840B.300d word vectors[2] to initialize them, and they are updated in the training process. Each POS token is represented by a one-hot vector, and updated during training too. This enables the model to learn the similarities between different POS tags (such as NNPS and NNS, for example). Table 1 gives a summary of input features.

Feature	Description
Word tokens	word embeddings of the mentions, and of 5 words before and after
POS tokens	part-of-speech tag embeddings
Speaker identity	whether two mentions are from the same speaker
Mention distance	how many other mentions are between them

Table 1: Input features

3.2 Triad Neural Network

Word embeddings are fed into a bidirectional LSTM layer, which generates a representation for each mention. The three members of the triad share the same LSTM layer. Similarly, POS embeddings are fed into a shared bidirectional LSTM layer.

$$h_i^{word} = \text{Word-LSTM}(X_i^{word}) \tag{2}$$

$$h_i^{pos} = \text{POS-LSTM}(X_i^{pos}) \tag{3}$$

where $i = 0, 1, 2$ is the index of the three mentions, X_i^{word} is the sequence of word embeddings used to represent mention i, and X_i^{pos} is the corresponding sequence of POS embeddings. Word-LSTM and POS-LSTM are both bidirectional, and shared by all input mentions. For each pair in the triad, the LSTM outputs for the two entities are concatenated with their joint features:

$$h_{ij} = f(X_{ij}^{speaker}, X_{ij}^{distance}, h_i^{word}, h_j^{word}, h_i^{pos}, h_j^{pos}) \tag{4}$$

[2]https://nlp.stanford.edu/projects/glove/

where $X_{ij}^{speaker}$ is a binary speaker identity feature for the mentions i and j, $X_{ij}^{distance}$ is the positive integer feature tracking the distance between them, and f represents several fully connected layer(s), shared by the three pairs. Our implementation uses two layers, with dropout between them.

While h_{ij} represents the relation between i and j, the other triad member needs taken into account as well. We achieve this by constructing a shared context h_{ijk}:

$$h_{ijk} = g(h_{ij} \bigoplus h_{jk} \bigoplus h_{ki}) \tag{5}$$

where g is another fully connected layer. Operator $\bigoplus$ means elementwise vector summation. Now, we can have a decoder layer d_{ij} for each of the pairwise relations.

$$d_{ij} = f_d(h_{ij}, h_{ijk}) \tag{6}$$

Function f_d is another fully connected layer. The three decoders work together to generate a 3D binary vector, as in equation 7. Each element represents whether the mention pair refers to the same entity (0 or 1).

$$\mathbf{y} = \text{sigmoid}(W(d_{ij}, d_{jk}, d_{ki}) + b) \tag{7}$$

where W and b are the weights and bias to be trained. The output $\mathbf{y}$ is a 1×3 vector. As we can see, the three decoders do not make decisions independently, but rather, "consult" with each other, as in equation 7. Each decoder also uses the shared context h_{ijk} at a lower level, as seen in equation 6.

3.3 Triads Generation

For n mentions in the text, there can be $\binom{N}{3}$ triads. In most cases, we have dozens of mentions in an article, which is not an issue. However, some long articles have hundreds of mentions, so generating all triads is unpractical and unnecessary. For instance, for 500 mentions, the total number of triads would be 20,708,500!

During the training process, we use only the mentions within the distance of 15 or less. In other words, we consider the pairs with 14 or fewer mentions between them. For testing, we consider the mentions with distances up to 40. However, this does not mean the long-distance coreference can never be detected. Often, coreferent mentions in-between the distant ones may serve as bridges, and our clustering algorithm is able to put them together. That being said, it is also true that long-distance mention pairs are less likely to corefer than those in closer proximity. Training with triads that include very distant pairs could also have the harmful effect of introducing too many negative samples.

3.4 Dyad Baseline System

To demonstrate that triads have advantages over a strictly pairwise approach, we also build a neural network model which takes mention pairs as input, and make binary decisions on the pairs only. The input features are the same as in the triad model, and the architecture can be considered a reduced triad system. Now there is no context information shared by three entities. The pair representation is directly connected to the output layer.

4 Entity Clustering

After the likelihood of pairwise coreference between all mentions has been determined, we use a clustering algorithm to group them. At the end of this process, each entity is represented by a mention cluster.

For every triad a, b and c, the system will produce three real values between [0,1] to represent the "probability" of a coreference link. We will refer to them as affinity scores. The higher the score, the more likely a pair of mentions refers to the same entity. The affinity score over a pair is computed as the average of their scores in all triads, as shown in equation 8.

$$\Phi(a, b) = \frac{1}{N} \sum_{c \in W(a,b)} \Phi(a, b; c) \tag{8}$$

Here, N is the total number of triads containing (a, b), $\Phi(a, b)$ is the affinity score of a and b, and $\Phi(a, b; c)$ is the affinity score of a and b when another mention c is in the triad. $W(a, b)$ represents the set of mentions within the distance window of a or b. We have experimented with other methods besides averaging, including taking the maximum, or the average of several top candidates. We found that the average produces better results.

The mutual distance between a and b is defined as the reciprocal of the affinity score, except we set the maximum value to be 10. Since the maximum value of $\Phi(a, b)$ is 1, the minimum value of $d(a, b)$ is also 1 according to equation 9. In principle, we would like distance metrics to have 0 as the minimum, which can be achieved by subtracting 1. However, for the purpose of clustering, it is not necessary.

$$d(a, b) = min\{\frac{1}{\Phi(a, b)}, 10\} \tag{9}$$

Recall that our system does not consider mention pairs too far apart in the text. For evaluation, the maximum distance for consideration is 40 (i.e. they may have up to 39 other mentions in between). We set the mutual distance between out-of-window mentions as 10, the maximal distance. As mentioned before, this does not mean they can never be clustered together. The result depends on the choice of linkage, and whether there is any coreferent entities in-between.

We use the hierarchical clustering function provided by SciPy library to build the sets of coreferent entities. Other than the customized distance metric, we used the default settings, opting for the $distance$, rather than $inconsistent$ cutoff criterion. When the clusters are built hierarchically, those with distances lower than a threshold are joined. Presumably, the threshold t should be lower than 2, which corresponds to affinity score higher than 0.5. The higher the threshold, the fewer clusters will be produced at the end. Preliminary experiments showed that the results are not affected very much when t is between 1.5$\sim$2.0. We used the $t = 1.7$, which corresponds to affinity score of 0.59.

The choice of linkage has a major impact on the results. We found the $single$ linkage produces the best results. Intuitively, it also makes sense. In a text, coreference among a group of mentions can only be recognized via one or several bridging mentions. In this case, the $single$ linkage can best represent the relation, while the $average$ and other linkages tend to overestimate the distances between clusters.

5 Experiments

For all the experiments, hyperparameters were tuned with the development set only. We use Adam optimizer with binary cross-entropy loss. The learning rate is initially set as 10^{-3}, then 5×10^{-4} after 100 sub-epochs, and 10^{-4} after 100 sub-epochs. We use the term "sub-epoch" to refer to training on 50 files, rather than the whole training set. The training set is relatively big, so we implemented a data generator with multiple subprocesses with a shared output queue. There are 1940 training files in total, so roughly all training files can be consumed in 40 sub-epochs, although smaller files may be used more frequently due to the nature of multiprocessing. The training completed in 300 sub-epochs. We use input dropout ratio 0.5 for word embeddings and POS embeddings. The last layer of each pair representations has dropout ratio 0.3. The bidirectional LSTMs use the average output of two directions.

For the baseline dyad model, the settings are similar. We also use 300 sub-epochs, but here it refers to training on 100 files, not 50.

5.1 Results of Triad Model

Table 2 shows the results of our triad system, compared to other systems in literature. All results are on the CoNLL 2012 English test data. As we can see, all the recent system have pretty much the same average F1 scores over the three metrics. Likely, they all capture the relatively easy cases, and the difficult ones remain to be tackled. Compared to other systems, ours has a very different distribution of scores across the three metrics, which suggests our results are quite different. Our system performs by far the best with the MUC evaluation metric, and is also the best with B^3 metric, measured with F1 score. However, the performance is quite low from the $CEAF_{\phi 4}$ metric. To understand the discrepancies, we need to analyze not only the nature of our system but also the nature of the metrics.

	MUC			B^3			$CEAF_{\phi4}$			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
Clark & Manning (2016)	79.91	69.30	74.23	71.01	56.53	62.95	63.84	54.33	58.70	65.29
Heuristic Loss	79.63	70.25	74.65	69.21	57.87	63.03	63.62	53.97	58.40	65.36
REINFORCE	80.08	69.61	74.48	70.70	56.96	63.09	63.59	54.46	58.67	65.41
Reward Rescaling	79.19	70.44	74.56	69.93	57.99	63.40	63.46	55.52	59.23	65.73
Triad	**84.93**	**77.26**	**80.92**	60.35	**71.77**	**65.65**	44.43	**59.20**	50.76	**65.78**

Table 2: Results of coreference resolution systems on the ConLL 2012 English test data. Our model (Triad) was trained on triads with the maximum distance of 15, making predictions on the triads with the maximum distance of 40. All other results are copied from Clark and Manning (2016).

MUC (Vilain et al., 1995) is a link-based metric. Mentions in the same entity/cluster are considered "linked". MUC penalizes the missing links and incorrect links, each with the same weight. It has been noted that MUC prefers over-merging entities (Luo, 2005) over under-merging. Incorrectly merging two entities is penalized less than incorrectly splitting an entity. Since we chose the $single$ linkage in our clustering algorithm, it is likely to over-merge sometimes.

B^3 (Bagga and Baldwin, 1998) is a mention-based metric. The evaluation score depends on the fraction of the correct mentions included in the response entities (i.e. entities created by the system). If a system does not make any decision and leaves every mention as singletons (i.e. no coreference at all), it will get a perfect precision score. Luo (2005) indicates that the B^3 precision score prefers no decision. On the other hand, the recall prefers over-merging entities. We have a high B^3 recall and relatively low precision, which reflects our results in MUC.

We manually spot-checked our results and the over-merging suggested by our MUC and B^3 scores appears to be true. Figure 1 provides an example to illustrate this. The top subfigure is the true clusters of entities, and the bottom shows the predictions. As we can see, the mentions are mostly grouped correctly, but there is a very large cluster (red) on the right, which basically merges two big ground truth clusters (light blue and yellow).

$CEAF_{\phi4}$ (Luo, 2005) assumes each key entity should only be mapped to **one** response entity, and vice versa. It aligns the key entities (clusters) with the response entities in the best way, and compute scores from that alignment. However, a major disadvantage of this approach is that the **unaligned responses are totally ignored**, even if they are legitimate clusters. Moosavi and Strube (2016) used an example to illustrate the problem. From a text, system $cr1$ identifies two entities $\{the\ American\ administration, it_1, it_2, it_3\}$ and $\{they_1, they_2, them, their\}$. However, it misses the fact that the two are actually the same entity. As we know, relatively few entities can be referred by both it and $they$, as $administration$ can. Another system $cr2$ only identifies the fist cluster, and misses all of the second cluster. Our intuition is $cr1$ does a better job, because it resolves much more coreference relations. However, $CEAF_{\phi4}$ will score $cr2$ higher. This is what happens to our system. There are 4217 true coreference entities from the test set. However, our system generated 5619 entities, or 33% more. Note our $CEAF_{\phi4}$ recall score is also 33% higher than $CEAF_{\phi4}$ precision score, which is related to their definitions. As a result, our system identifies a lot of small subsets of entities, but may miss some crucial links in between. MUC and B^3 will give partial credits to it, but $CEAF_{\phi4}$ completely ignores the efforts and only picks one subset for the best alignment.

In order to understand the distribution of entity/cluster sizes, we collected all the entities from test data and from our system response. Figure 2 shows the distribution of entities with respect to their size. As we can see, our system generates some very large entities, quite a few bigger than 100. This is evidence of over-merging. On the other hand, the system also generates many small entities, as shown by the left-most green bar. Note the y-axis there is in log scale, so the difference is not proportionally visualized. Figure 1 also shows that there are 13 true entities in the file, but the system finds 17. On the one hand, it merges two major entities; but on the other hand, it breaks down some small entities.

Different evaluation metrics help to diagnose different problems, and every system, as well as every

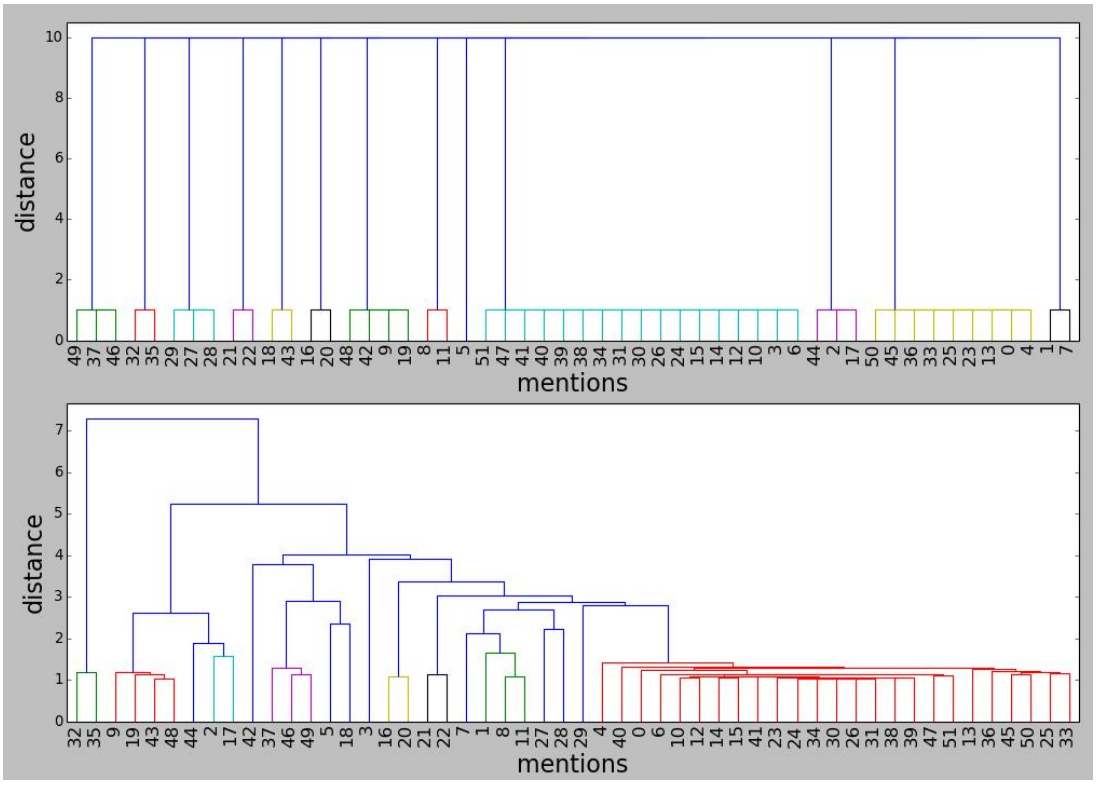

Figure 1: Clustering results from the test file voa_0049. Top subfigure is the true clusters. Bottom is the predicted results. Mentions with the same color are in the same cluster. The cutoff threshold is 1.7.

metric, would have its strength and weakness. In practice, what is needed depends on the purpose. CoNLL 2012 Shared Task uses the average of the three metrics to rank systems. Even though our system shows the best performance, the average here may not be very meaningful, given that our system has a very different distribution of scores from the others.

5.2 Results of Dyads Model

The results of the dyad system are shown in Table 3. As we can see, the triad system has a big advantage over the dyad system. Although we use maximum mention distance 15 for training, for prediction, it is beneficial to allow larger distances in the triad model. However, even in prediction, it is difficult to increase the distance beyond a certain point, since the number of triads increases very fast when the allowed distance becomes larger. Below, we show the results of both dyad and triad systems using the distance of up to 15 and up to 40 at test time.

	MUC			**B^3**			**$CEAF_{\phi4}$**			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Dyad, max distance 15	76.83	78.16	77.49	46.03	70.59	55.73	44.43	41.58	42.96	58.73
Dyad, max distance 40	77.80	**82.84**	80.24	37.65	**77.86**	50.76	**47.99**	36.49	41.46	57.48
Triad, max distance 15	84.48	75.26	79.60	**62.36**	68.30	65.20	42.78	**59.98**	49.94	64.91
Triad, max distance 40	**84.93**	77.26	**80.92**	60.35	71.77	**65.65**	44.43	59.20	**50.76**	**65.78**

Table 3: Results of the dyad model compared to the triad model. Two maximum mention distances are tested: 15 and 40. They are all trained with maximum distance 15.

Generally, we found that dyad-based system generates more entities (less coreference) than the triad system. For test data, we had 11,299 entities from the dyad-based model, but only 5,619 from the triad-based model.

Triad model can also support additional restrictions. For example, we can require at least one pair in a

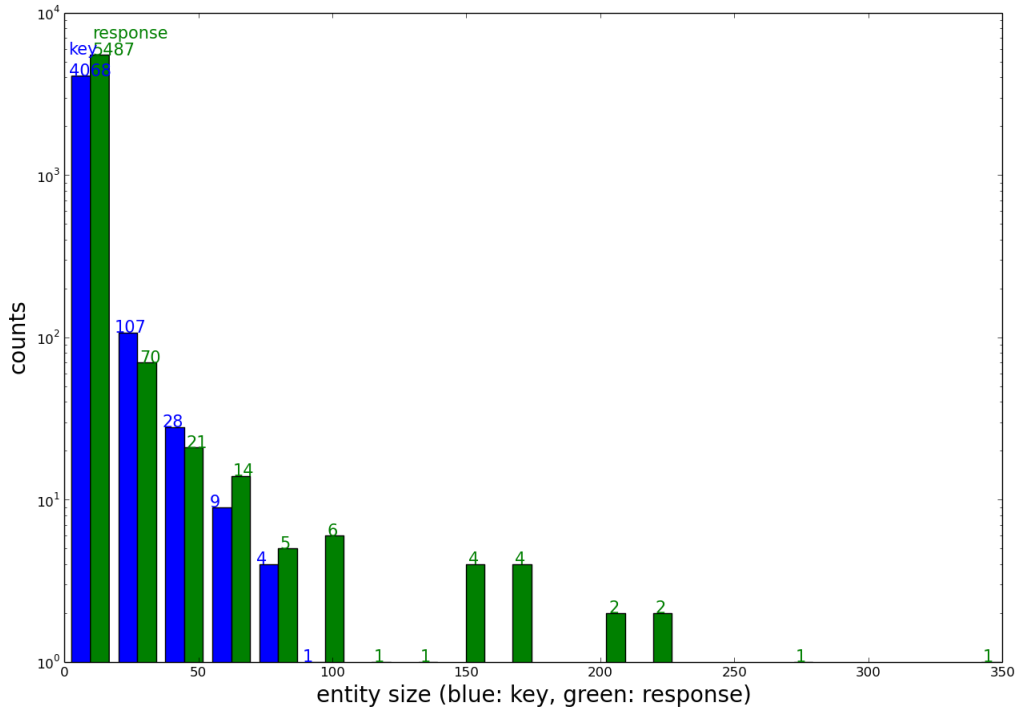

Figure 2: Entity sizes (number of mentions). Horizontal axis is the size of entities. Blue/left bars are the true counts from test set. Green/right bars are the counts from system response. Note the vertical axis is drawn in log scale. Compared to the truth, our system produces more extreme cases i.e. entities with very small number of mentions, or very big number of mentions.

triad to have a smaller distance. The point of allowing longer distances between mentions is to identify coreferent mentions that are far apart in text. However, it is typically fairly rare to have mentions that are far away refer to the same entity. We do not have to allow all sides of a triangle to be big, and imposing this restriction may improve the overall quality of the response entities.

Note that this system can be easily extended from triads to tetrads (union of four mentions) and higher polyads. Sometimes we may want to look at two more other places to determine whether a coreference relation is present. Ideally, the larger the polyad, the better we can capture mutual dependencies. However, since the number of polyads grows fast with the polyad order, the computation may quickly become intractable for larger texts.

6 Conclusion

We developed a triad-based neural network model that assigns affinity scores to mention pairs. A standard clustering algorithm using the resulting scores produces state-of-art performance on MUC and B^3 metrics. Our system appears to behave quite differently from others, judging by its performance on different metrics. A dyad-based baseline model has substantially lower performance, suggesting that using triads plays an important role. Note that approaches other than clustering, such as the mention ranking models, can easily be used with our output as well, and we expect some of them would work better than the simple agglomerative clustering.

Mutual dependencies among multiple mentions are important in coreference resolution tasks, but it is often ignored. Our triad-based model addresses this gap. This model can be additionally constrained to improve performance, and if necessary, easily extended from triads to polyads with higher order.

Acknowledgments

This project is funded in part by an NSF CAREER award to Anna Rumshisky (IIS-1652742).

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *CoRR*, abs/1609.08667.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, October. Association for Computational Linguistics.

David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. 2018. Neural relational inference for interacting systems. *CoRR*, abs/1802.04687.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, August.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27(4):521–544, December.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *HLT-NAACL*, pages 994–1004. The Association for Computational Linguistics.