Neural Coreference Resolution

Kevin Clark
Department of Computer Science
Stanford University
Stanford, CA 94305
kevclark@cs.stanford.edu

Abstract

Much work on coreference resolution has gone towards hand crafting complicated features that are predictive of coreference. However, systems relying on these can become unwieldy and may generalize poorly to new data. In this work, I present a new coreference system based on neural networks that automatically learns dense vector representations for mention pairs. These representations are built almost entirely from distributed representations of the words in the mentions and surrounding context, and can capture semantic similarity important for coreference. I use these representations to train an incremental coreference system that can exploit entity-level information. Despite using a minimal number of hand-engineered features, the neural coreference system is competitive with the current state of the art on the OntoNotes dataset.

1 Introduction

Coreference resolution is the task of identifying mentions in a text that refer to the same real-world entity. For example, the mentions "Obama", "the president", and "he" could all refer to Barack Obama. Coreference resolution is considered an important aspect of text understanding, and has numerous applications from information extraction to question answering to machine translation.

Most coreference resolution systems rely on complex highly engineered features capturing relevant syntactic, semantic, and discourse-level information [3, 35, 30] and some are even entirely rule based [29]. However, recent work by Durrett and Klein [11] demonstrates that lexical features generated from simple templates can be very effective for coreference. In contrast to using a large set of sparse lexical features, this work investigates using dense features from distributed work representations instead. These can provide more efficient learning and better generalization because similar words have similar vector representations.

The distributed word representations are used to train deep neural networks for coreference. Deep learning models have been successful for a variety of natural language processing tasks [8, 33, 19], and can capture much more complicated interactions than the linear models commonly used for coreference resolution. To the best of my knowledge, this is the first time deep learning has successfully been applied to the task of coreference resolution.

I first describe a neural mention pair model that predicts whether two mentions are coreferent or not and then discuss extending this model by untying the weight matrices depending on the types of the involved mentions, which allows the model to better distinguish between the linguistically different phenomena of nominal coreference resolution and pronominal anaphora resolution. Using the mention pair model as a building block, I then propose a way of training an entity-centric coreference system that learns an effective policy for building up coreference chains incrementally. This system has the ability to let previous coreference decisions to inform later ones through entity-level information shared with a max pooling operation.

I evaluate the models experimentally on the OntoNotes coreference dataset. Despite using very small number of hand engineered features, the neural coreference systems are competitive with recent state-of-the-art results. Furthermore, the incremental coreference system significantly outperforms the mention pair model with simple best-first clustering, showing it is able to effectively combine the information from many mention pairs when making decisions. Error analysis shows that the deep learning models are able to capture semantic similarity between mentions, an aspect of coreference that previously has been considered and "uphill battle" in coreference resolution [11].

2 Mention Pair Model

Mention pair models perform binary classification, predicting whether two mentions belong in the same coreference cluster or not. Mention pair models have historically been very effective for coreference [34, 26, 3, 35, 4] and when combined with a secondary clustering model can achieve state-of-the-art results [7]. My neural mention pair model first builds up a vector representing the pair of mentions in question. This vector is then passed through a feed-forward neural network to produce a prediction.

2.1 Features

Pairs of mentions are represented with features from each of the mentions and their contexts as well as features capturing relations between the pair.

Mention Features:

- Word vectors for the first, last, and head words of the mention.
- The average of all word vectors in the mention.
- The 10 features used by Rescanes et al. (2013) [31] for singleton mention identification, which capture aspects of a mention like the gender, animacy, quantification, and negation.
- The dependency relation between the head word of the mention and its parent (binary feature on the 10 most common dependency relations).

Context Features:

- The average of the word vectors of all words to the left (or right) of the mention in a window of 1, 5, or infinity (e.g., the entire sentence to the left of the mention).

Mention-Relating Features:

- The distance between the mentions in number of sentences and number of intervening mentions. Distances are binned (e.g., one bin might denote a sentence distance between 5 and 10) and then encoded as a 1-hot vector.
- Two binary speaker identification features classifying the mentions as either being almost certainly coreferent or incompatible for coreference (e.g., "you" and "me" from the same speaker are incompatible).
- Three binary string matching features indicating if the mentions have matching head words, match partially, or match exactly.

I used 50 dimensional word embeddings from `word2vec` [24] trained on the Wikipedia and Gigaword corpus for these features. Currently these vectors are treated as static and are not updated during training. In the spirit of deep learning minimizing the need for hand-engineered features, I also evaluate the model when only using distance features and features from word vectors.

2.2 Neural Network Architecture

Let m_i and c_i denote that mention features and context features for mention i and $r_{(i,j)}$ denote the mention pair features for two mentions i and j. Given two mentions i and j with i preceding j in the text, the neural mention pair model first concatenates the features from the two mentions to produce

a single vector representing the pair. This vector is then passed through a standard neural network with two hidden layers:

$$\begin{aligned}
\textbf{Input Vector Construction:} \quad & x = [m_i, m_j, c_i, c_j, r_{(i,j)}] \\
\textbf{Hidden Layer 1:} \quad & h^{(1)} = \texttt{ReLU}(W^{(1)}x + b^{(1)}) \\
\textbf{Hidden Layer 2:} \quad & h^{(2)} = \texttt{ReLU}(W^{(2)}h^{(1)} + b^{(2)}) \\
\textbf{Prediction Layer:} \quad & p(i,j) = \texttt{sigmoid}(w^T h^{(2)})
\end{aligned}$$

The final output $p(i, j)$ is taken to be the probability that mentions i and j are coreferent. The network is trained by finding parameters θ that minimize the negative log likelihood of the data:

$$\mathcal{L}(\theta) = -\sum_{i,j \in \mathcal{M}, i<j} \log(x_{i,j}p(i,j) + (1 - x_{i,j})(1 - p(i,j)))$$

Where $\mathcal{M}$ is the set of all mentions in the document and $x_{i,j}$ is a boolean variable that takes value 1 if mentions i and j are coreferent and 0 if otherwise.

Untying the Weight Matrices based on Mention Type The linguistic phenomena in coreference resolution can vary significantly based on the"types" of the involved mentions. In particular, the important information necessary for pronominal anaphora resolution and nominal coreference resolution are extremely different. For example, string matching and measures for semantic similarity are powerful features for nominal coreference resolution, but are not applicable for pronominal anaphora resolution. To capture this, I explored using different weight matrices for the first layer based on the types of the involved mentions. Let t_i denote the "type" of mention i: either nominal or pronominal. With untied weight matrices, the first hidden layer computes:

$$h^{(1)} = \texttt{ReLU}(W_{t_i}^{(m_1)}m_i + W_{t_j}^{(m_2)}m_j + W^{(c)}[c_i, c_j] + W_{t_i,t_j}^{(r)}r_{(i,j)} + b^{(1)})$$

Thus the model essentially learns two matrices for (for example) the mention features for the first mention $W^{(m_1)}$. Although I could have gone further by also untying the context weight matrix, or by creating more mention types such as separating proper nouns from common nouns, this would significantly increase the number of parameters in the model, and I found no improvement from doing this in preliminary experiments. Figure 1 shows the full neural network architecture with untied weights.

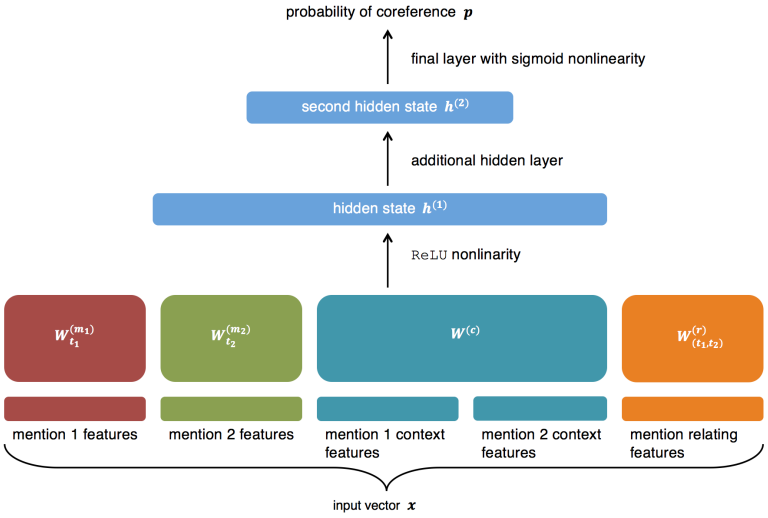

Figure 1: Mention pair model with untied weights.

Training details: The network was regularized using dropout [16] with a rate of 0.5. During training, the loss was minimized with AdaGrad [10] for 50 epochs with minibatches of size 200. The model was evaluated on a held out dataset after each epoch and the highest scoring parameters were selected for the final model. I trained the model with 300 units in the first hidden layer and 100 units in the second one.

3 Incremental Coreference Model

Mention pair scores alone are not enough to produce a final set of coreference clusters because they do not enforce transitivity: if the pair of mentions (a, b) and the pair of mentions (b, c) are deemed coreferent by the model, there is no guarantee that the model will also classify (a, c) as coreferent. Thus a second step is needed to coordinate the scores to produce a final coreference partition. There have been many different proposed ways of doing this [18, 34, 26, 14, 23, 27], but all of these approaches have the weakness of only relying on local (pairwise) information to make decisions. This means the systems cannot consolidate information at the entity level. As a result, coreference chains produced by such algorithms can exhibit low coherency. For example, a cluster may consist of [Hillary Clinton, Clinton, he] because the prediction between *Hillary Clinton* and *Clinton* is made independently of the one between *Clinton* and *he*.

A more sophisticated approach is to forgo using a mention pair model and instead train a classifier that operates between two *clusters* of mentions instead of two mentions. This has the advantage of allowing earlier coreference decisions to inform later ones; for example finding *Clinton* and *he* are coreferent makes it less likely that *Hillary Clinton* and *Clinton* are coreferent.

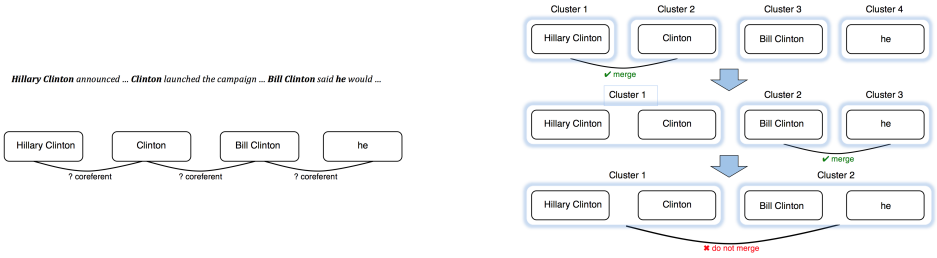

Figure 2: Left: mention pair model, right: incremental coreference model

3.1 Test Time Inference

At test time, the classifier acts as an agent that builds up coreference chains with agglomerative clustering. It begins in a start state where each mention is in a separate single-element cluster. At each step, it observes the current state, which consists of all partially formed coreference clusters produced so far, and selects some action which merges two existing clusters. The action will result in a new state with new candidate actions and the process is repeated.

The number possible cluster merges is quadratic in the number of clusters, so considering all of them would make inference very slow. To deal with this, I took the approach of Clark and Manning [7] and pruned the search space with scores produced from the mention pair classifier. First, the agent orders all mention pairs in the document in descending order according to their pairwise scores. This causes clustering to occur in an easy-first fashion, where harder decisions are delayed until more information is available. Secondly, the agent discards all mention pairs that score below a threshold t under the assumption that the clusters containing these pairs are unlikely to be coreferent. Algorithm 1 shows the full test-time procedure.

3.2 Learning

We face a sequential prediction problem where future observations (visited states) depend on previous actions. This is challenging because it violates the common i.i.d. assumptions made in statistical learning. For example, naively training the agent on the gold labels alone would unrealistically teach the agent to make decisions under the assumption that all previous decisions were correct, potentially causing it to over-rely on information from past actions. This is especially problematic in coreference, where the error rate is quite high

Imitation learning, where expert demonstrations of good behavior are used to teach the agent, has proven very useful in practice for this sort of problem [1]. I use imitation learning to train the agent to classify whether an action (merge or do not merge the current pair of clusters) matches an expert policy. In particular, I use the DAgger imitation learning algorithm [32]. Costs are added to actions

Algorithm 1 Inference method: agglomerative clustering

Input: Set of mentions in document $\mathcal{M}$, pairwise classifier with parameters $\boldsymbol{\theta}_p$, agent with parameters $\boldsymbol{\theta}_a$, cutoff threshold t
Output: Clustering C

Initialize list of mention pairs $P \rightarrow []$
for each pair $(i, j) \in \mathcal{M}^2$ with $i < j$ **do**
  if $p_{\boldsymbol{\theta}_p}(i, j) > t$ **then**
    P.append((i, j))
  end if
end for
Sort P in descending order according to $p_{\boldsymbol{\theta}_c}$

Initialize $C \rightarrow$ initial clustering with each mention in $\mathcal{M}$ in its own cluster
for $(i, j) \in P$ **do**
  if $C[i] \neq C[j]$
  and $p_{\boldsymbol{\theta}_a}(C[i], C[j]) > 0.5$ **then**
    DoMerge$(C[i], C[j], C)$
  end if
end for

based on the B^3 coreference resolution metric [2], which gives the model a concept of the severity of a mistake. This is important for learning because some cluster merges (e.g., between two very large clusters) impact the score much more than others. The full learning procedure is the same as the one used by Clark and Manning [7], which is described in detail in their paper.

The important information for the purpose of this work is that DAgger assembles a dataset that is used to train the agent. Each example in this dataset consists of two clusters (c_{i_1}, c_{i_2}) and a cost s_i which provides a measure of how beneficial it is to merge the two clusters. A negative cost means the merge is good (i.e., will lead to a high scoring coreference partition) and a positive cost means the merge is bad. The goal is now to train a binary classifier that assigns high probability to pairs of clusters with low costs and low probability to pairs of clusters with high costs.

3.3 Neural Network Architecture

A neural network is trained to do this task. The classifier builds a representation of a pair of clusters using the pairs of mentions that exist between the two clusters (i.e., each mention is in a different cluster). For each pair of mentions, it computes a feature vector using the same architecture as the mention pair model except it stops after producing $h^{(2)}$ and returns it. The model then applies a max-over-time feature pooling operation [8] to these vectors across all mention pairs. The full architecture is shown in Figure 3.

Building Mention Pair Representations:	for each mention pair $(m_i, m_j) \in c_1 \times c_2$ compute a vector representation $v_{(m_i,m_j)}$ the same way as computing $h^{(2)}$ in the neural mention pair model
Building a Cluster Pair Representation:	$v_{(c_1,c_2)} = \max\{v_{(m_i,m_j)} : (m_i, m_j) \in c_1 \times c_2\}$
Prediction Layer:	$p(c_1, c_2) = \texttt{sigmoid}(w^T v_{(c_1,c_2)})$

The network is trained by finding parameters θ that minimize the risk over the dataset:

$$\mathcal{L}(\theta) = \sum_i s_i p_\theta(c_{i_1}, c_{i_2})$$

where s_i is the cost of merging clusters c_{i_1} and c_{i_2}.

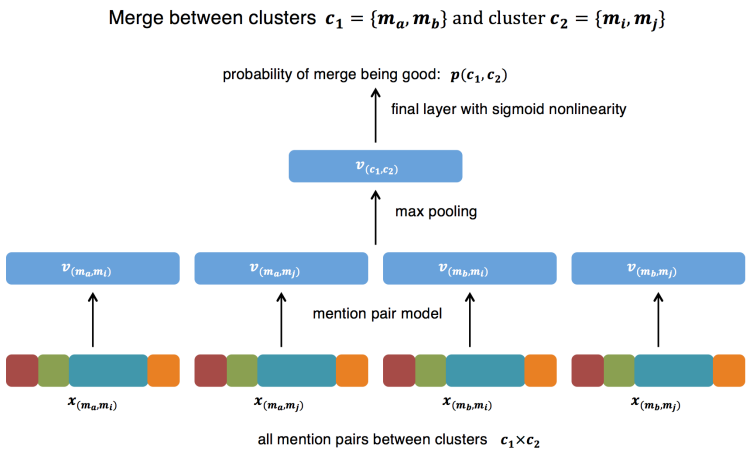

Figure 3: Mention pair model with untied weights.

4 Related Work

There has been extensive work on mention pair models for coreference [34, 26, 3, 35, 4], but none of these use distributed word representations or neural network models. Most rely on complex hand-crafted engineered features, and analysis of coreference resolution features has been done in great depth [3, 35, 20, 31].

In contrast, Durrett and Klein [11] propose a coreference model using a much simpler set of sparse lexical features from simple templates. Although this does greatly reduce the hand-engineering going into the coreference system, sparse lexical features lack the shared representation that makes distributed word embeddings so powerful. Their system also operates entirely on mention pairs, so it cannot capture entity level information the way an incremental coreference model can.

Incremental coreference systems that can use entity-level information have also been explored by previous work. One approach is using mention-entity models that go through mentions left to right, assigning each mention to a (partially completed) cluster [21, 37, 30]. Our system, however, builds clusters incrementally through merge operations, and so can operate in an easy-first fashion, delaying later decisions until more information is available. Raghunathan et al. [29] take this approach with a rule-based system that runs in multiple passes and Stoyanov and Eisner [36] train a classifier to do this with a structured perceptron algorithm. These systems all consolidate entity-level information using hand built rules. In contrast, the neural incremental model *learns* how mention-pair features should be combined at the entity level.

Imitation learning has been employed to train coreference resolvers on trajectories of decisions similar to those that would be seen at test-time by Daumé et al. [9] and Ma et al. [22]. This work differs from these in that it makes use of a cost function instead of only learning which actions look good locally according to a heuristic.

This work is most similar to Clark and Manning's [7] system, which learns an incremental policy for building up coreference chains in the same way as this system's. However, that work creates features between clusters of mentions from the pairwise probabilities produced by mention pair models (for example, one feature is the maximum probability of coreference over the mention pairs between the two clusters). In contrast to only using a single score to produce cluster-level features, this work combines all the features from vector representations of mention pairs.

I am not aware of any end-to-end coreference system using deep learning. Guha et al. [15] use distributed word representations in their mention pair coreference model, but only use them to produce a cosine similarity score as a feature in a logistic regression classifier. The most similar deep learning work to his one is perhaps Chen and Manning's research on neural dependency parsing [6], in which the model also learns which local actions lead to a desirable state. However, because the

error rate for systems in dependency parsing is much lower than for coreference, they do not use an imitation learning algorithm for training.

5 Experiments

5.1 Experimental Setup

Dataset: I run experiments on data from the English portion of the CoNLL 2012 Shared Task [28], which is derived from the OntoNotes corpus [17]. The dataset consists of about 4500 news articles annotated with gold standard coreference clusters. The data comes already split into training, dev, and test sets.

Mention Detection: All experiments were run using system-produced predicted mentions. I used the rule-based mention detection algorithm from Raghunathan et al. [29], which first extracts pronouns and maximal NP projections as candidate mentions and then filters this set with rules that remove spurious mentions such as numeric entities and pleonastic *it* pronouns. Using a simple high-recall but low-precision approach like this is common for mention detection. The system can later learn to ignore incorrect mentions produced in this step by not assigning them to a coreference cluster.

5.2 Mention Pair Classification

Training set construction: Large documents have a huge number of mention pairs, but only a small fraction of these will be coreferent. To reduce the class imbalance and large size of the dataset, I down-sampled negative to produce more manageable sets for training. Most experiments are run using a training set of 2.5 million mention pairs, but I also evaluate the final model on a larger set of 7.5 million pairs.

Evaluation: I evaluate the mention pair binary classifiers using F_1 score and area under the precision-recall curve (auc). Examples from different documents are grouped together in this evaluation (as opposed to computing the average F_1 for a document). Doing this is common for coreference evaluation because it stops small documents from having an overly large influence.

Results: Experimental results for mention pair classification on the development set are in Table 1. I first explore the benefits of untying the weight matrices in the first layer of the mention pair model. Doing this resulted in an improvement of 0.6 F_1 and 0.004 auc, showing there are gains from separating the learning for pronominal anaphora and nominal coreference resolution.

One advantage of the deep learning approach to coreference is the lack of hand-engineered features going into the system. I evaluated the importance of the few hand-engineered features included in the model with an ablation study, training models without any hand-crafted features or with only the 5 string matching and speaker identification features. The hand-engineered features proved to be quite important, causing an almost 5 point improvement in accuracy. Most of this gain, however, comes from the speaker identification and string matching features.

I also compare with the state of the art mention pair model from Clark and Manning's [7] coreference system. This model is a logistic regression classifier with over one hundred syntactic, semantic, lexical, and distance-based features in addition to a complicated feature conjunction scheme. Despite using a much simpler feature set, the best neural network model outperforms this classifier when trained on datasets of the same size.

Error Analysis Although overall performing about the same as Clark and Manning's system, the neural mention pair model performs about 3 F_1 points better on proper-nominal coreference (56.3 F_1 vs 53.5). Some example mention pairs that the neural mention model gets right that Clark and Manning's system does not includes (*Cuba*, *the island*), (*the USS Cole*, *the ship*), (*Christ*, *the Son of God*), (*Bush*, *the president*). Even without the context of the surrounding text, these mentions are clearly likely to be coreferent because they are semantically similar. However, capturing this kind of similarity in automatic coreference models has long been elusive [11]. Word embeddings have been known to capture this sort of semantic information [25], and these wins suggest that neural network approaches to coreference can make headway in incorporating shallow semantics into coreference resolution systems.

Model, Training Set Size	F_1	**auc**
Pairwise model from Clark and Manning (2015), 32M (entire training set)	64.0	0.696
Pairwise model from Clark and Manning (2015), 2.5M	62.0	0.675
Neural mention Pair Model, 2.5M	63.0	0.679
without untied weights	62.4	0.674
with only speaker identification and string matching hand-engineered features	61.5	0.667
with no hand-engineered features	58.4	0.609
Neural mention Pair Model, 7.5M	64.0	0.689

Table 1: Mention Pair classification accuracies on the development Set

	MUC			B^3			$CEAF_{\phi_4}$			CoNLL
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Avg. F_1
Fernandes et al.	75.91	65.83	70.51	65.19	51.55	57.58	57.28	50.82	53.86	60.65
Björkelund and Kuhn	74.3	67.46	70.72	62.71	54.96	58.58	59.4	52.27	55.61	61.63
Ma et al.	81.03	66.16	72.84	66.90	51.10	57.94	68.75	44.34	53.91	61.56
Durrett and Klein	72.61	69.91	71.24	61.18	56.43	58.71	56.17	54.23	55.18	61.71
Clark and Manning	76.12	69.38	72.59	65.64	56.01	60.44	59.44	52.98	56.02	63.02
Mention Pair Model	73.84	66.93	70.22	61.12	54.14	57.41	54.44	51.31	52.83	60.15
Incremental Model	77.08	67.79	72.13	66.03	54.37	59.63	58.48	52.28	55.21	62.32

Table 2: Comparison of this work with other state-of-the-art approaches on the test set.

5.3 End to End Coreference

Evaluation: The models are evaluated using three of the most popular metrics for coreference resolution: MUC, B^3, and Entity-based CEAFE ($CEAF_{\phi_4}$). We also include the average F_1 score (CoNLL F_1) of these three metrics, as is commonly done in CoNLL Shared Tasks

Results: In Table 3 we compare the results of our system with the following state-of-the-art approaches: Clark and Manning [7], the Berkeley system [12]; the Prune-and-Score system [22]; the HOTCoref system [5]; and Fernandes et al.[13]. Despite using a small number of hand-crafted features, the model is competitive to the current state-of-the art, outperforming all recent works except for Clark and Manning's. The model does particular well with the B^3 metric, which is unsurprising because these are used to produce costs during training.

In addition to evaluating the incremental coreference system, we also evaluate the mention pair model with best-first clustering [26], which assigns mentions the highest scoring previous mention as the antecedent. This coreference system performs significantly worse (over 2 points in CoNLL F_1), showing that the neural network coreference model benefits greatly from incorporating entity-level information when making decisions.

6 Conclusion

I introduced a new approach to coreference resolution that uses distributed word representations in neural network models to make predictions. These models use significantly fewer hand-crafted features then current state-of-the-art systems, but remain competitive with them in performance. The neural mention pair model performs on par with the complicated mention pair model from Clark and Manning [7], and outperforms it on proper-nominal resolution, where it can make better model semantic similarity important for coreference. We also show untying the weight matrices of the mention pair model to distinguish pronominal anaphora and coreference resolution resulted in gains in accuracy. Using the mention pair model as a component piece, I describe a incremental coreference system that operates between clusters of mentions instead of pairs, which allows previous coreference decisions to inform later ones. This approach significantly outperforms the mention pair model with simple best-first clustering, suggesting it is able effectively to exploit entity level information.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

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

[3] Eric Bengtson and Dan Roth. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303, 2008.

[4] Anders Björkelund and Richárd Farkas. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 49–55, 2012.

[5] Anders Björkelund and Jonas Kuhn. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Association of Computational Linguistics (ACL)*, 2014.

[6] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP*, pages 740–750, 2014.

[7] Kevin Clark and Chris Manning. Entity-centric coreference resolution with model stacking. In *Association of Computational Linguistics (ACL)*, 2015.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[9] Hal Daumé III and Daniel Marcu. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–104, 2005.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[11] Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1971–1982, 2013.

[12] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics (TACL)*, 2:477–490, 2014.

[13] Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 41–48, 2012.

[14] Jenny Rose Finkel and Christopher D Manning. Enforcing transitivity in coreference resolution. In *Association for Computational Linguistics (ACL), Short Paper*, pages 45–48, 2008.

[15] Anupam Guha, Mohit Iyyer, Danny Bouman, Jordan Boyd-Graber, and Jordan Boyd. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *HLT-NAACL*, 2015.

[16] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[17] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 57–60, 2006.

[18] Andrew Kehler. Probabilistic coreference in information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–173, 1997.

[19] Yoon Kim. Convolutional neural networks for sentence classification. *Empirical Methods in Natural Language Processing (EMNLP*.

[20] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, 2011.

[21] Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Association for Computational Linguistics (ACL)*, page 135, 2004.

[22] Chao Ma, Janardhan Rao Doppa, J Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich, and Prasad Tadepalli. Prune-and-score: Learning for greedy coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[23] Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 905–912, 2005.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[25] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 746–751, 2013.

[26] Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 104–111, 2002.

[27] Cristina Nicolae and Gabriel Nicolae. Bestcut: A graph algorithm for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 275–283, 2006.

[28] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 1–40, 2012.

[29] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501, 2010.

[30] Altaf Rahman and Vincent Ng. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research (JAIR)*, pages 469–521, 2011.

[31] Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. The life and death of discourse entities: Identifying singleton mentions. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 627–633, 2013.

[32] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 627–633, 2011.

[33] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP*.

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

[35] Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. Reconcile: A coreference resolution research platform. 2010. Computer Science Technical Report, Cornell University, Ithaca, NY.

[36] Veselin Stoyanov and Jason Eisner. Easy-first coreference resolution. In *COLING*, pages 2519–2534, 2012.

[37] Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. An entity-mention model for coreference resolution with inductive logic programming. In *Association of Computational Linguistics (ACL)*, pages 843–851, 2008.