# Toward Featureless Event Coreference Resolution via Conjoined Convolutional Neural Networks

**Anonymous ACL submission**


## Abstract

Coreference resolution systems for entities and/or events almost always make use of many linguistic, parsing-based features. In contrast, (1) we introduce a new state-of-the-art event coreference resolution system which uses only lemmatization and its corresponding precomputed word/character embeddings, and we exhaustively illustrate the performance of other commonly-used features. (2) The crux of our system is that it first makes pairwise event-coreference predictions by using a Conjoined Convolutional Neural Network. Last, (3) we cluster event mentions with a novel neural approach which performs merges in an easy-first, cluster-holistic manner, allowing our system to be less susceptible to errors that are made exclusively from min-pairwise decisions. When performed on a common test set of the ECB+ corpus, our system achieves CoNLL F1 scores of 67.2 and 59.3 for within-document and cross-document tasks, respectively.


## 1 Introduction

Coreference resolution is the task of trying to identify – within a single text or across multiple documents – which *mentions* refer to the same underlying discourse *entity* or *event*. A *mention* is defined as a particular instance of word(s) in a document (e.g., *she* or *announced*). Ultimately, coreference resolution is a clustering task, whereby we wish to group all like-mentions together, as shown in Figure 1. Successfully doing so can be useful for several other core NLP tasks that concern natural language understanding, such as information extraction (Humphreys et al., 1997), topic detection (Allan et al., 1998), text summarization (Daniel et al., 2003), (Narayanan and Harabagiu, 2004), etc.

> Lindsay Lohan checked into California's Betty Ford Center Rehab Facility late Thursday, dodging arrest ...
>
> Prior to this check-in to Betty Ford, which was first reported by tmz.com, California prosecutors had said ...
>
> The actress had checked-out of a similar facility, according to spokesman ...

Figure 1: Sample of the ECB+ corpus, depicting gold coref mentions as having shared box colors.

Specifically, coreference systems aim to find a globally-optimal fit of mentions to clusters, whereby every mention $m$ in the corpus is assigned to exactly one cluster $C$, the membership of which constitutes that every $m_i, m_j \in C_k$ is co-referent with each other. If a given $m_i$ is not anaphoric with any other $m_j$, then it should belong to its own $C_k$ with a membership of one. Further, the number of distinct clusters is not known apriori, and it is implicitly the responsibility of the system to determine. Finding a globally-optimal assignment of clusters is NP-Hard and thus computationally intractable. In attempt to avoid this, systems typically perform pairwise-mention predictions, then use those predictions to build clusters. The specific modelling strategies for such approximately fall into two categories: (1) mention-ranking / mention-pairs; and (2) entity-level / event-level.

**Mention-ranking** models define a scoring function $f(m_i, m_j)$ which operates on a mention $m_j$ and possible antecedent $m_i$, where $m_i$ occurs earlier in the document and could be null (represented by $\epsilon$ and denoting that $m_j$ is non-anaphoric). Wiseman, et. al.'s (2016a) strong model is an example.

**Mention-pair** models are defined almost identically to mention-ranking ones, with the subtle difference being the target objective of the pairwise-candidates. That is, mention-ranking model aim to find the ideal $m_i$ antecedent for every $m_j$, whereas mention-pair models score all possible $(m_i, m_j)$ pairs. Because these models base their predictions on the information from just two mentions at a time, they are by definition less expressive than entity/event-level models. However, their inference can be relatively simple and effective, allowing them to be fast and scalable. Consequently, they have often been the approach used by many state-of-the-art systems (Soon et al., 2001; Durrett and Klein, 2013).

**Entity/Event-level** models differ in that they focus on building a global representation of each underlying entity or event, the basis of which determines each mention's membership – as opposed to operating on a mention-level basis (Wiseman et al., 2016a; Clark and Manning, 2016b).

In this work, we use a novel mention-pair model that is designed to discriminate between pairs of input features: Siamese Convolutional Neural Networks. We feel awkward using the term "siamese," so we will henceforth refer to our model as our newly-coined term, Conjoined Convolutional Neural Networks (or CCNN). Further, we aim to replace a common weakness of mention-pair models with an approach resembling the main strength of entity/event-level models. Specifically, we aim to combine all linked mention pairs into a cluster via a neural, easy-first clustering approach which factors in a small, but effective, notion of the entire cluster at large.

Additionally, coreference research systems typically use a plethora of relatively expensive parsing-based features, including dependency parse information, lemmatization, WordNet hypernyms/synonyms, FrameNet semantic roles, etc. Although some research papers list their system's learned feature weights (Yang et al., 2015), there has been a striking lack of work which takes the minimalist approach and illustrates the effects of using few features. We aim to address this by starting with a basic lemma-embedding feature and then evaluate on our dev set the effectiveness of adding other commonly used features.

In summary, we are interested in within-document and cross-document *event* coreference, which has received drastically less attention than *entity* coreference. We introduce a novel mention-pair approach, using a Conjoined Convolutional Neural Network and unusually few features. We contribute detailed results of other commonly used features. And last, we combine our predicted mention pairs into clusters via a simple neural approach which represents each cluster as a whole, yielding state-of-the-art results on the ECB+ corpus.

## 2 Related Work

The seminal research on event coreference can be traced back to the DARPA-initiated MUC conferences, whereby the focus was on limited scenarios involving terrorist attacks, plane crashes, management succession, resignation, etc. (Humphreys et al., 1997; Bagga and Baldwin, 1999).

In the present day, Deep Learning is revolutionizing NLP; however, there has been only a few successful applications of deep learning to coreference resolution, almost all of which have been for *entity* coreference. We attribute this dearth to the fact that coreference resolution is inherently a clustering task, which tends to be a non-obvious modality for deep learning. Since our model (1) uses deep learning and (2) operates on the ECB+ corpus, we organize the related research accordingly.

### 2.1 Deep Learning Approaches

To the best of our knowledge, there are only five publications which apply deep learning to coreference resolution, four of which focus on entity coreference.

Sam Wiseman, et. al. (2015; 2016b) trained a mention-ranking model with a heuristic loss function that assigns different costs based on the types of errors made, and their latter work used mention-ranking predictions towards an entity-level model. Clark and Manning (2016b; 2016a) also built both a mention-ranking model and an entity-level model, the former of which was novel in using reinforcement learning to find the optimal loss values for the same four distinct error types defined in Wiseman's, et. al. (2015) work.

### 2.2 Systems using ECB+ Corpus

For our research, we make use of the ECB+ corpus (Cybulska and Vossen, 2014), which we further describe in Section 3.4. This rich corpus provides annotations for both entities and events, yet most research chooses to focus on using *either* events *or* entities, not both. To the best of our knowledge, there are only two papers which focus on the event mentions of ECB+: The Hierarchical Distance-dependent Chinese Restaurant Process (HDDCRP) model by Yang, et. al. (2015) and Choubey's and Huang's Iteratively-Unfolding approach (2017).

#### 2.2.1 HDDCRP Model

Yang, et. al's HDDCRP model (2015) uses a clever mention-pair approach, whereby they first use logistic regression to train parameters $\theta$ for the similarity function in Equation 1.

$$f_\theta(x_i, x_j) \propto \exp\{\theta^T \psi(m_i, m_j)\} \qquad (1)$$

Then, in a Chinese-restaurant-process fashion, they probabilistically link together mentions based purely on the scores provided by this similarity function. That is, the value of $f(m_i, m_j)$ is directly correlated with the probability of $(m_i, m_j)$ being chosen as a linked pair. Then, identical to Bengtson's and Roth's work (2008), the HDDCRP model forms clusters by tracing through all linked pairs. All mentions that are reachable by a continuous path become assigned the same cluster. This hinges on the transitive property of coreference. For example, if $(m_1, m_3), (m_3, m_5)$ and $(m_5, m_6)$ are each individually linked via the scoring function, then a cluster $C_i$ is formed, where $C_i = \{m_1, m_3, m_5, m_6\}$, even though $(m_1, m_5)$ or $(m_3, m_6)$ may have had very low similarity scores. We aim to improve this shortcoming, as detailed in Section 5.

#### 2.2.2 Neural Iteratively-Unfolding Model

Recently, Choubey and Huang (2017) introduced the first neural model for event coreference. Their system also fits into the mention-pair paradigm, whereby mentions are predicted by a feed-forward neural network. Their model features are based on the cosine similarity and Euclidean distance of input-pair embeddings. The authors asserted that when using the ECB+ corpus, within-doc coreference did not benefit from using mention context, which is an important finding. However, similar to the weakness of the HDDCRP model, they merge clusters which contain *any* mention-pair whose predicted score is below a given threshold, independent of mentions' relation to the cluster at large.

## 3 System Overview

### 3.1 Mention Identification

Coreference systems are predicated upon having entity/event mentions identified. In fact, this identification process is the focus of a different line of research: entity recognition and event detection. This separation of tasks allows coreference systems to be evaluated precisely on their ability to cluster together appropriate mentions, independent from the mention detection. Thus, it is common practice for coreference systems to either: (1) use an off-the-shelf entity recognition system, or (2) use gold mentions that are defined by the true annotations in the corpus. To illustrate the effectiveness of our system, we do both, and we use the exact same set of mentions as each system we compare against:

The **HDDCRP** model chose to not use the gold test mentions provided by ECB+. Instead, they used a semantic role labeller to identify and filter all test mentions. To compare against their system, we worked with the HDDCRP source code and data to ensure we accurately use their same mentions. Independent of clustering performance: of the 3,290 gold mentions in ECB+, their system failed to identify/use 676 of them. Of their 3,109 predicted mentions, 495 were false positives and in fact not actual gold mentions. Since their system uses these imperfect mentions, yet tests the coreference performance against the gold mentions, their system encounters a steep performance loss by default. With complete confidence, we fairly compare our system with theirs by using the same, imperfectly-predicted mentions.

The Iteratively-Unfolding system also aimed to use the same mentions as HDDCRP's. After numerous exchanges with the author, it was clear that their set of *test* mentions was similar and reasonable for research, but understandably not the same as that used by HDDCRP. Namely, they filtered out: (1) all predicted mentions which were not in the gold set (false positives), and (2) predicted mentions which did not cluster with a mention from another document. When we compare against this system, we followed their convention.

Lastly, to show an upper-bound of performance, we also report results having trained and tested our system using the gold mentions that are identified in the ECB+ corpus.

| | Train | Dev | Test | Total |
|---|---|---|---|---|
| # Documents | 462 | 73 | 447 | 982 |
| # Sentences | 7,294 | 649 | 7,867 | 15,810 |
| # Mentions-1 | 1,938 | 386 | 2,837 | 5,161 |
| # Mentions-2 | 142 | 52 | 240 | 434 |
| # Mentions-3 | 18 | – | 25 | 43 |
| # Mentions-4 | 6 | – | 7 | 13 |

Table 1: Statistics of the ECB+ Corpus, where Mentions-N represents event mentions which are N-tokens in length.

### 3.2 Reproducibility

We provide our code online,[1] which is written in Keras (Chollet, 2015) and is easily runnable on any of the aforementioned sets of mentions and evaluations. Additionally, our code runs in just a few minutes on a single Titan X GPU.

### 3.3 Models

Our coreference systems are comprised of two neural models:

- Conjoined Convolutional Neural Network (CCNN) – used for making mention-pair predictions. (Section 4)
- Neural Clustering (NC) – uses CCNN's pairwise predictions to cluster mentions into events (Section 5)

Further details about how these models are used together are detailed in Section 6.

### 3.4 Corpus

We use the ECB+ corpus (Cybulska and Vossen, 2014), which is the largest available dataset with annotations for event coreference. The corpus is comprised of 43 distinct *topics*, totaling 982 documents. We maintain the same train/dev/test splits as previous researchers, as further detailed in Section 6. A sample of the corpus in shown in Figure 1, and statistics are listed in Table 1, where it is clear that the majority of gold mentions are one token in length (e.g, *announced*).

## 4 Conjoined Convolutional Neural Network (CCNN)

### 4.1 Motivation

In the training data, the mentions that belong to the same gold cluster often have little variance amongst their text. This, coupled with Choubey's, et. al. (2017) conclusion that context does not improve within-doc performance for our corpus, lends us to believe that an event-level model would be unnecessary and probably worse than a mention-pair model. Thus, we continue the trend of using a mention-pair model for event coreference.

[1] WILL_INSERT_GITHUB_LINK

### 4.2 Overview

Conjoined Neural Networks (a.k.a. Siamese Networks) were first introduced by Bromly and LeCun (1994) for the task of determining if two input items (hand signatures) were from the same person or not. Specifically, a Conjoined Network can be defined as two identical neural networks, each of which accepts distinct inputs, which are joined by a single loss function over their highest-level features. The loss function computes a similarity score (e.g., euclidean distance) for an input pair. The two networks are said to be conjoined because they share the same weights and thus work together as one network that learns how to discriminate. The benefits of tying the weights are that it: (1) ensures that similar inputs will be mapped appropriately, otherwise, they could be mapped to hidden representations that are disproportionately dissimilar from their input representations; and (2) forces the network to be symmetric. Namely, if we were to abstractly view the Conjoined Network as a function, then:

$$CCNN(f_i, f_j) \equiv CCNN(f_j, f_i)$$

This is critical, as the CCNN's similarity function should be independent of the ordering of its input pair.

Last, Conjoined Networks have been shown to perform well in low-resource situation (Gregory Koch, 2015). This is ideal for our task, as it is highly likely that at test time we will encounter event mentions that are OOV. We desire our model to discriminately learn the relationships of input mentions, rather than exclusively relying on and memorizing the input values themselves. Likewise, we choose to use a Convolutional Network due to (1) their power in learning sub-regions of features and the relations thereof, and (2) their recent advances in many NLP tasks, including sentence classification (Kim, 2014), machine translation (Gehring et al., 2017), dependency parsing (Yu and Vu, 2017).

### 4.3 Input Features

Since our CCNN needs each mention to be represented exclusively by its own input, we used none of the relational features[2] that are common in other coreference systems (e.g., SameLemma, Jaccard similarity of mentions' context, shared WordNet parents). We used Stanford CoreNLP (Manning et al., 2014) to extract the following features, which we thoroughly tested in different ways:

- **Part-of-Speech:** LSTM-learned POS embeddings; and 1-hot representations.
- **Lemmatization**: Lemmatized each token and represented it by pre-trained GloVe (Pennington et al., 2014) word embeddings.[3]
- **Dependency Lemma:** we represent the dependent parent/children of each token via their aforementioned lemma embeddings.
- **Character Embeddings:** each token is represented as a concatenation of its character embeddings.
- **Word Embeddings:** pre-trained GloVe word embeddings.

We account for mentions' having varying token lengths by summing their tokens in place, thus representing each mention as a fixed-length vector.

### 4.4 Architecture

We define the full embedding for a given token $t$ as $t_{emb} = t_{f_1} \oplus t_{f_2} \oplus \ldots \oplus t_{f_n}$, where $\oplus$ represents vector concatenation and $t_{f_i}$ represents a specific input feature vector.

Naturally, we may want to convolve over the context of mention $m$, too, by including the $N$ words before and after $m$. Thus, our input for mention $m$ is a matrix $M$, and a la Kim (2014), we zero-pad unfilled windows.

Let $\mathbf{M}$ represent the full matrix corresponding to mention $m$: $\mathbf{M} \in \mathbb{R}^{(2N+1)\times d}$ and $\mathbf{M}_{(i,j),(k:l)}$ represent the sub-matrix of $M$ from $(i, j)$ to $(k, l)$.

We define a kernel with dimensions $(h, w)$, where $h < (2N + 1)$ and $w < d$. This allows the kernel to operate on sub-sections of the embeddings. The kernel has an associated weight matrix $\mathbf{w} \in \mathbb{R}^{h\times w}$. Starting at a given index $(i, j)$ within mention matrix $\mathbf{M}$, a feature $c_i$ is defined as:

$$c_i = f(\mathbf{w}^T \mathbf{M}_{(i:i+h-1),(j:j+w-1)} + b) \qquad (2)$$

where $b \in \mathbb{R}$ is an added bias term. The kernel runs over every possible sub-section of mention matrix $\mathbf{M}$, yielding a feature map $\mathbf{c} \in \mathbb{R}^{(2N-h)\times(d-w-1)}$

We start with 64 kernels and ReLU as our activation function. We apply dropout and repeat the process once more. Next, we apply max-pooling to get a single $\hat{c} = max\{\mathbf{c}\} \in \mathbb{R}$ for each kernel. Last, we merge all kernels' univariate vectors into 1 final layer, then we use ReLU to yield a two-class softmax prediction. Since the network is comprised of two identical, conjoined halves, we sufficiently illustrate only one half in Figure 2. Complete parameter values such as dropout and kernel sizes are listed in our Supplemental Materials document.

### 4.5 Loss / Optimization

Our goal is to maximize discriminability of different event mentions, while enforcing features to be as similar as possible when they are of the same event. Contrastive Loss, shown in Equation 3, is perfectly suited for this objective (Schroff et al., 2015; Liu et al., 2016). Our training set has a strong class imbalance (most input pairs are not co-referent), so we down-sample to a 5:1 ratio of negative-to-positive examples. We use Adagrad for optimization.

$$\begin{aligned} L(\hat{y}, y) = \frac{1}{2N} \sum_{n=1}^{N} [(y)d^2 \\ + (1 - y) * (max(1 - d, 0))^2] \\ \text{where } d = \|y_n - \hat{y}_n\|_2 \end{aligned} \qquad (3)$$

## 5 Neural Clustering (NC)

It is common practice for mention-pair models to first assign a probability score to every mention-pair, and then cluster with a different model.

### 5.1 Existing Clustering Approaches

**Agglomerative Clustering** is a simple but effective approach. It first assigns each mention to its own singleton cluster. Then, it repeatedly merges the two distinct clusters which contain the

[2] We experimented with extending our CCNN model by adding relational features as a merged-layer at the highest neural level. However, doing so provided no benefit.

[3] 300-dimensions; built from an 840 billion token crawl.

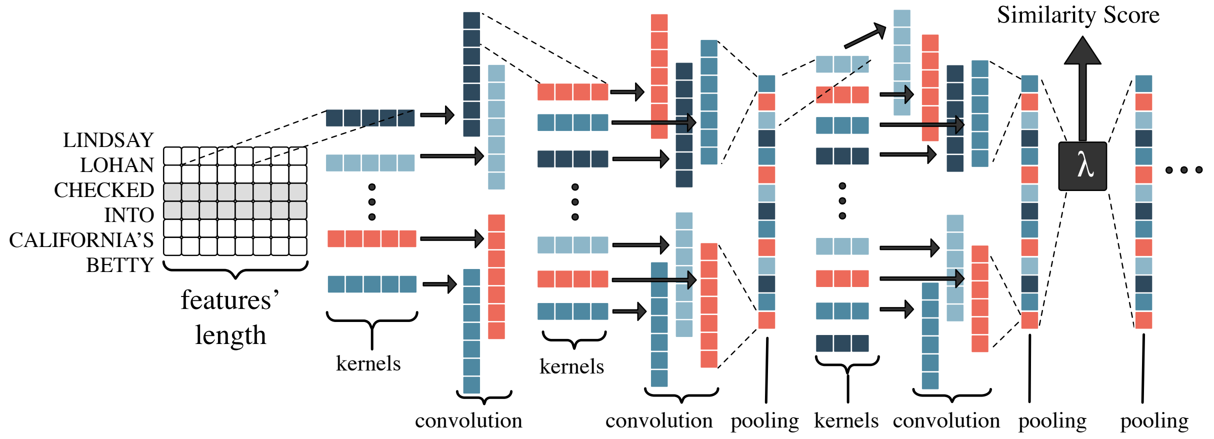


Figure 2: One half of the Conjoined Convolutional Neural Network's Architecture. The Lambda function represents the Euclidean Distance of the merged univariate vectors from the identical networks halves.

shortest-distance mention pairs. Although this is a strong baseline, as seen in Yang, et. al. (2015), there are three main weaknesses:

1. One must define a stopping threshold $\alpha$.
2. Any given $\alpha$ hinges on the data being uniform across documents. In reality, distances between mention-pairs could vary significantly between documents and topics.
3. Most significant, each cluster merge is based solely on two individual mentions, yet these mentions may not be representative of the cluster at large.

**HDDCRP and Iterative-Folding Clustering** both contain the issue #3 from above, as detailed in Sections 2.2.1 and 2.2.2.

### 5.2 Our Approach

We aim to use the strengths of agglomerative clustering, while replacing its shortcomings. We train a classifier to learn the most likely positive cluster merge, where the cluster is represented more holistically than a mention-pair basis.

Specifically, we learn a function $f(C_x, C_y)$ that predicts the likelihood of an appropriate, positive merging of clusters $(C_x, C_y)$. Let $d(m_i, m_j)$ be the mention-pair distance predicted by our CCNN model, where $m_i \in C_x$, and $m_j \in C_y$. Function $f(C_x, C_y)$ is based on four simple features:

- min-pair distance: $\min_{m_i,m_j} d(m_i, m_j)$
- avg-pair distance: $\frac{\sum_{m_i,m_j} d(m_i,m_j)}{\|C_x\|\|C_y\|}$
- max-pair distance: $\max_{m_i,m_j} d(m_i, m_j)$
- size of candidate cluster: $\frac{\|C_x\|+\|C_y\|}{\sum_z \|C_z\|}$

The first three features serve to better represent the cluster at large (issue #3 from above). For example, a given cluster $C_1$, when evaluated against two other candidate clusters $C_2$ and $C_3$, may have the same minimum mention-pair distance score with both $C_2$ and $C_3$ Yet, the average and maximum distance scores shed more light onto which cluster has more similar mentions. The last feature (cluster size) represents the size percentage of our considered merge, relative to all mentions in our current set. This serves is an explicit measure to help prevent clusters from growing too large, and is not as vulnerable to issue #2.

### 5.3 Architecture

We define $f$ as a feed-forward neural network[4] which predicts the probability of a positive cluster merge, via a two-class softmax function. Our loss function is weighted binary cross-entropy, to account for the class imbalance situation that most pairs of clusters should not be merged together.

### 5.4 Inference

Our system will incrementally build up clusters, starting with each cluster having just one mention (in the within-document scenario). Thus, it is important to train our Neural Clustering model on positive and negative examples of clusters in varying states of completeness. Our gold truth data in-

[4] We used 1 hidden layer of 25 units, ReLU activation without dropout, and Adagrad as our optimizer.

forms us which mentions are co-referent, but since there is no single canonical ordering in which mentions should become co-referent, we generate synthetic data to represent possible positive and negative examples of when clusters should be merged.

Specifically, for training, we generate a positive example by randomly sampling a golden cluster, followed by splitting the cluster into two random subsets. The above four features are calculated for these two subsets of clusters, and the target output is a positive case. Likewise, we generate negative examples by sampling random subsets from disjoint golden clusters.

Again, at test time, we use Neural Cluster to evaluate every possible $(C_x, C_y)$ cluster pair in an easy-first manner. That is, at each iteration, we merge only the $(C_x, C_y)$ pair that yielded the highest score (likelihood of a positive merge). Then, we re-evaluate all pairs with our newly merged cluster, and repeat this process until the model no longer predicts any merge. Thus, unlike the aforementioned models, we do not *require* an additional parameter for our stopping criterion.

## 6 Our Coreference Systems

We use our CCNN and Neural Clustering (NC) models together to perform coreference resolution. The only differences between the within-document and cross-document scenarios are our data and evaluation metric, as described below.

### 6.1 Training / Development / Testing Data

We adhere to the same data splits as previous researchers, whereby the dev set is topics 23-25, and the test set is topics 26-45. Traditionally, topics 1-22 are used as training. However, since our NC model relies on our CCNN's predictions, we remove topics 19-22 from the training set and instead use them as dev sets for our NC models. The complete details are provided in our Supplemental Materials document.

### 6.2 Within-Document

We train a CCNN model on mention-pairs which appear in the same document, and using its predictions on a held-out set, we train the NC to predict when to merge clusters.

### 6.3 Cross-Document

Cross-document resolution is a superset of the within-document task; it concerns all coreference chains, regardless if mentions in a cluster were originally from the same document or not. Our cross-document system is identical to our within-document one, but: (1) we train a separate CCNN only on mention-pairs which are from different documents; (2) instead of initializing our clustering with each mention serving as a singleton cluster, we use our within-document NC predictions as starting clusters; (3) at each iteration, we only consider merging clusters $(C_x, C_y)$ if $C_x$ and $C_y$ contain mentions from disjoint sets of documents. Thus, our cross-document NC only uses cross-document mention pairs distances for its decisions. Since we will never merge two within-doc clusters from the same document, we rely on our within-document NC to accurately form zero or one clusters for each unique event.

## 7 Results

As a recap, our research concerns three independent axis of investigation:

- **Features:** which features are most useful, and can we use few features?
- **Mention-Pair Model:** how well does CCNN perform against a standard feed-forward neural network[5] (FFNN)?
- **Clustering:** can we outperform Agglomerative via our Neural Clustering model?

Our metric is CoNLL F1 score, which is a clustering-based metric that combines the F1 scores of MUC, $B^3$, and $CEAF_e$, and we use the official scorer script (v8.01) (Pradhan et al., 2014).

We were interested in the five common, non-relational embedding features which are detailed in Section 4.3: POS, Lemma, Dependency Lemma, Character, Word. We tested all combinations of features on the Dev Set, and Lemma + Character Embedding yielded the best dev results (see Figure 3). Thus, our CCNN + Neural Clustering system used only these two features in its evaluation against other systems, as illustrated in Table 2. The results show that our CCNN model outperforms a FFNN, and that our Neural Clustering outperforms Agglomerative Clustering. Further, when training and testing on gold mentions, we achieved CoNLL F1 scores of **81.2** and **72.4**

[5]Given two mentions $i$ and $j$, with corresponding feature vectors $f_i$ and $f_j$, their input to the Feed-Forward Neural Network is the vector $\|f_i - f_j\|$

for within-document and cross-document, respectively. We denote these scores as the new baseline to which to compare future systems.

**Lemma Embeddings** were the most useful feature, followed closely by Character Embeddings. Since *SameLemma* has historically been a strong baseline, this is unsurprising.

**Character Embeddings** were effective for the same reason String Edit Distance is often a strong feature: there tends to be a direct correlation between the textual similarity of mentions and their likelihood of being co-referent. Both random character embeddings and pre-trained ones yielded the same performance, suggesting that the power comes from the uniqueness of characters, not any *meaning* conveyed in the characters.

Empirically, Lemma + Character Embeddings are complementary features; the semantic information conveyed within the lemma embeddings complement the syntactic information of character embeddings. Related, POS by itself was a poor feature, but combining it with either Lemma or Character Embeddings offered strong results.

Ideally, a classifier should precisely learn how to combine all features such that the unhelpful features are given no weight. However, in practice, that is often extremely difficult, due to both the large parameter space and the high entropy wherein some combinations of features seem to equally help as much as hurt. Thus, we conclude that one should try to use the fewest features as possible for coreference resolution, then expand appropriately.

Interestingly, in all experiments, our results were inversely correlated with the amount of context our CCNN used. That is, our best performance came when we used no context, only the mention words. This agrees with the Choubey's, et. al. findings (2017).

### 7.1 Comparison to Other Systems

*SameLemma* has historically proven to be a strong baseline. That is, anytime two mentions have identical lemmas, simply mark them as being coreferent.

Using the same mentions that were used by the HDDCRP and Choubey (Iteratively-Unfolding) systems, our flagship CCNN+NC system yielded the highest results, despite using few features.

## 8 Conclusion

In summary, we researched event coreference resolution, and our approach was novel by using a Conjoined Convolutional Neural Network as a mention-pair model, followed by a Neural Clustering model. Unlike most coreference systems which rely on dozens of features, we used only lemmatization and pre-trained word/character embeddings. We illustrate the performance of five commonly used features, demonstrating the effectiveness of using few features. Our Neural Clustering model addressed a common weakness in mention-pair models: instead of forming clusters based on just *one* mention-pair satisfying a criterion, we used features which represent clusters more holistically. On a common test set, our system achieved state-of-the-art performance with CoNLL F1 scores of 67.2 and 59.3 for within-document and cross-document, respectively. Further, when using gold mentions, we set a new within-document baseline of 81.2 and cross-document baseline of 72.4.

## References

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, James Allan Umass, Brian Archibald Cmu, Doug Beeferman Cmu, Adam Berger Cmu, Ralf Brown Cmu, Ira Carp Dragon, George Doddington Darpa, Alex Hauptmann Cmu, John Lafferty Cmu, Victor Lavrenko Umass, Xin Liu Cmu, Steve Lowe Dragon, Paul Van Mulbregt Dragon, Ron Papka Umass, Thomas Pierce Cmu, Jay Ponte Umass, and Mike Scudder Umass. 1998. Topic detection and tracking pilot study final report. In *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218.

Amit Bagga and Breck Baldwin. 1999. Cross-document event coreference: Annotations, experiments, and observations. In *Proceedings of the Workshop on Coreference and Its Applications*, CorefApp '99, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In J. D. Cowan, G. Tesauro, and J. Alspector,

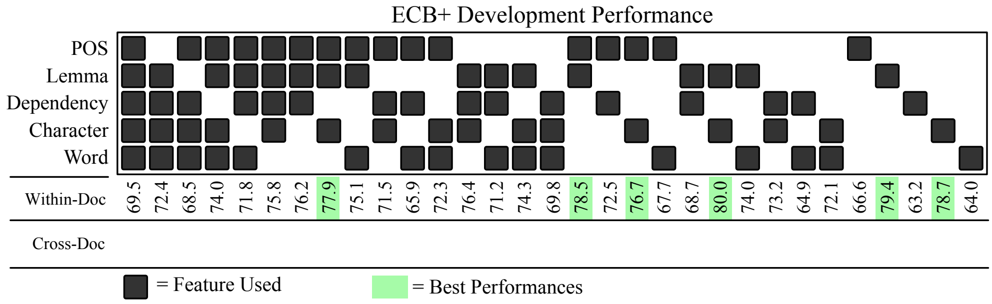


Figure 3: The CoNLL F1 performance of our flagship CCNN + Neural Clustering system, using all combinations of features.

| | Within-Document | | | | Cross-Document | | | |
|---|---|---|---|---|---|---|---|---|
| | MUC | $B^3$ | CEAF | CoNLL F1 | MUC | $B^3$ | CEAF | CoNLL F1 |
| Test Set: ECB+ Gold Mentions | | | | | | | | |
| SameLemma | 58.3 | 83.0 | 75.9 | 72.4 | - | - | - | - |
| FFNN+AGG | 59.9 | 85.6 | 78.4 | 74.6 | - | - | - | - |
| FFNN+NC | 60.7 | 86.7 | 79.4 | 75.6 | - | - | - | - |
| CCNN+AGG | 70.5 | 89.1 | 83.5 | 81.0 | - | - | - | - |
| **CCNN+NC** | 70.9 | 88.9 | 83.6 | **81.2** | - | - | - | - |
| Test Set: HDDCRP's Predicted Mentions | | | | | | | | |
| SameLemma | 40.4 | 66.4 | 66.2 | 57.7 | - | - | - | - |
| HDDCRP | 53.4 | 75.4 | 71.7 | 66.8 | - | - | - | - |
| **CCNN+NC** | 53.7 | 75.2 | 71.9 | **66.9** | - | - | - | - |
| Test Set: Choubey's et. al. Mentions | | | | | | | | |
| SameLemma | 48.8 | 66.7 | 65.1 | 60.2 | - | - | - | - |
| Choubey | 62.6 | 72.4 | 71.8 | 68.9 | - | - | - | - |
| **CCNN+NC** | 67.3 | 73.0 | 69.5 | **69.9** | - | - | - | - |

Table 2: Comparison against other systems, while our models use only the Lemma + Character Embedding features. FFNN denotes a Feed-Forward Neural Network Mention-Pair model. AGG denotes Agglomerative Clustering.

editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan-Kaufmann.

Franois Chollet. 2015. keras. `https://github.com/fchollet/keras`.

Prafulla Kumar Choubey and Ruihong Huang. 2017. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *EMNLP*.

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

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. Cite arxiv:1606.01323Comment: Accepted for publication at the Association for Computational Linguistics (ACL), 2016.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Naomi Daniel, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC '03, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*, pages 1971–1982. ACL.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *ACL (1)*, pages 123–135. Association for Computational Linguistics.

Ruslan Salakhutdinov Gregory Koch, Richard Zemel.

2015. Siamese neural networks for one-shot image recognition. In *ICML*.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, ANARESOLUTION '97, pages 75–81, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751. ACL.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 507–516, New York, New York, USA. PMLR.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society.

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.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016a. Learning global features for coreference resolution. *CoRR*, abs/1604.03035.

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

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL (1)*, pages 1416–1426. The Association for Computer Linguistics.

Bishan Yang, Claire Cardie, and Peter I. Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *TACL*, 3:517–528.

Xiang Yu and Ngoc Thang Vu. 2017. Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. *CoRR*, abs/1705.10814.