Towards Featureless Event Coreference Resolution via Conjoined Convolutional Networks

Chris Tanner and **Eugene Charniak**
Brown Linguistic Laboratory of Information Processing
Brown University
Providence, RI 02912
{christanner,ec}@cs.brown.edu

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-/char- embeddings, achieving 67.2 CoNLL F1 score on a common ECB+ test set, along with setting a new baseline of 8X.XX for the test set at large. (2) We exhaustively illustrate the performance of other commonly-used features. 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 the event mentions with a simple, but 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.

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*. Naturally, one may be solely interesting in determining if two given entities co-refer to the same object (e.g., a pairwise prediction of *she* and *Mary* co-referring); however, 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), knowledge base population (Mayfield and et al., 2009), question answering (Narayanan and Harabagiu, 2004), etc.

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

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 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 up clusters. The specific modelling strategies for such approximately fall into two categories: (1) mention-ranking / mention-pairs; and (2) entity/event-level.

Mention-ranking models define a scoring a 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 ϵ 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, 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. Although these models are by definition less expressive than entity/event-level models, 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, which, for political reasons, we will henceforth refer to 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, best-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, part-of-speech, etc. Although some research includes a listing of their system's learned feature weights (Yang et al., 2015), there has been a striking lack of work which takes the minimalistic 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 the effectiveness of slowly adding other commonly used features.

Finally, in general, *event* coreference resolution has received drastically less attention than *entity* coreference. However, in this paper, we are exclusively interested in event coreference.

In summary, we introduce a novel mention-pair approach to event coreference resolution by 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 attempts to represent each cluster as a whole, yielding us with 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 present day, Deep Learning is revolutionarily affecting 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 into categories 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) built mention-ranking models which are trained with a heuristic loss functions that assign different costs based on the types of errors made, and their latter work used mention-ranking predictions towards an entity-level model via LSTM hidden states (Hochreiter and Schmidhuber, 1997).

Clark and Manning (2016b; 2016a) also built both a mention-ranking 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), an extension of EventCorefBank (ECB) (Bejan and Harabagiu, 2010), 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 θ 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 emitted by $f(m_i, m_j)$ 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 (Bengtson and Roth, 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 holding true for 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 within-doc model's features are primarily based on the cosine similarity and euclidean distance of input-pair embeddings. The cross-document model is identical, other than adding context features, too. This was an important finding, for they assert that when using the ECB+ corpus, within-doc coreference did not benefit from using mention context. Although, similar to the weakness of the HDDCRP model, they form clusters based on local mention-pair predictions, independent of mentions' relevance 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 gold mentions that are defined by the true annotations in the corpus, or (2) use an off-the-shelf entity recognition system. We do both. That is, the majority of our results are shown with having used gold ECB+ test mentions. Yet, to ensure we developed a competitive system, it was imperative that we compare our system to the HDDCRP and Iterative-Unfolding models.

The HDDCRP model used a pre-existing event detection system to predict mentions, then they filtered many of those, yielding their system with an imperfect but reasonable set of mentions that shares a moderate overlap with the gold test mentions. Determining the exact mentions that were used by HDDCRP was one of the most challenging and time-consuming processes of our research.

Naturally, the Iteratively-Unfolding system also aimed to use their same mentions. After numerous exchanges with the author, it was clear that their set of 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 were singletons (ones that did not cluster with a mention from another document).

We evaluate our systems having used the: (1) gold mentions; (2) HDDCRP-predicted mentions; (3) Choubey-predicted mentioned.

3.2 Reproducibility

We provide our code online[1], which 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 system is comprised of two neural models:

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

[1] www.github.com/

	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 mentions which are N-tokens in length

3.4 Corpus

We exclusively make use of 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* – categories or news stories – each of which has 2 sub-topics which are similar in nature but distinctly different from each other. For example, Topic 1 contains two sub-topics: one about Lindsay Lohan checking into a rehab center, the other about Tara Reid doing so. Each sub-topic contains roughly 10 short text documents which all concern the same given sub-topic. We maintained the same train/dev/test splits as previous researchers[2] 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

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 within-doc coreference.

4.2 Overview

Conjoined Neural Networks (a.k.a. Siamese Networks) were first introduced by Bromly and LeCun (1994) towards a task of determining if two input items (hand signatures) were in fact of the same class or not. Specifically, a Conjoined Network can be defined as twin 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, we posit that CCNN's 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.

As for the choice of Conjoined Network, Convolutional Neural Networks (CNNs) have recently proven to be highly useful for many tasks in NLP, including sentence classification (Kim, 2014), machine translation (Gehring et al., 2017), dependency parsing (Yu and Vu, 2017), etc. Likewise, we choose to use a CNN due to their power in learning sub-regions of features and the relations thereof.

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 that are common in other coreference systems (e.g., SameLemma, Jaccard similarity of mentions' context, first common WordNet parent, etc). We used Stanford CoreNLP (Manning et al., 2014) to extract the following features, which we thoroughly tested in different ways:

[2] Training set contains topics 1-20, the dev set contains topics 21-23, and the test set has topics 24-43.

- **Part-of-Speech:** LSTM-learned POS embeddings; and 1-hot representations.
- **Lemmatization**: 300-length word embeddings for the lemma of each token, where the embedding came from running GloVe (Pennington et al., 2014) on our corpus, or using their provided pre-trained 6 billion and 840 billion token crawls.
- **Dependency Lemma:** we use the dependent parent and/or children of each token, and we represent it via the aforementioned lemma embeddings.
- **Character Embeddings:** we represent each token as a concatenation of its character embeddings, where we experimented with 20-dimensional character embeddings being either random or pre-trained.
- **Word Embeddings:** the same embeddings that we use for the lemma feature, only the token is represented by its actual word, not lemma.

Note, since mentions are of varying token length (see Table 1), we need a convention to standardize the vector-length (e.g., to 300 dimension). We experimented with summing all token embeddings in place, averaging, and concatenated to a particular N-length size. For clarity, *averaging* for a given mention m's embedding is calculated via: $m_{emb}[i] = \frac{\sum_{t \in T} t_{emb}[i]}{|T|}$, where t is a token in the set of tokens T that comprise m.

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 use 64 kernels (we experimented with powers of 2, from 2-256), along with ReLU as our activation function. Dropout of 0.1 is then applied. Next, we repeat this processing by adding convolution and dropout again, then we apply max-pooling to get a single $\hat{c} = max\{\mathbf{c}\} \in \mathbb{R}$ for each kernel. Last, we merge all of our kernels' univariate vectors into 1 final layer, apply dropout again, and feed into ReLU which yields a two-class softmax prediction. The full architecture is shown in Figure 4.4.

4.5 Loss

Our goal is to maximize discriminability between mentions representing different events, while enforcing features to be as similar as possible when they are of the same event. Contrastive Loss is perfectly suited for this objective (Schroff et al., 2015; Liu et al., 2016), as shown in Equation 3. 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.

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

$$\text{where } d = \|a_n - b_n\|_2$$

4.6 Optimization

We used Adagrad for optimization. RMS yielded effectively identical results.

5 Neural Clustering

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 treats each mention as being its own singleton-cluster. Then, it repeatedly merges the two distinct clusters which contain the 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 heuristic.
2. If one uses a threshold value α to stop (i.e., merge while the shortest-pair is $< \alpha$), this relies on the data being uniform across documents. Yet, the distribution of pairwise predictions inevitable differs for each document, causing any given α to be appropriate for some documents but sub-optimal for others.
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 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 the shortcomings. We train a classifier to learn the most likely positive cluster merge, where the cluster is represented more holistically than a single pair. Instead of operating on a mention-to-mention basis to dictate the cluster merges, we consider every possible mention-to-cluster.

Specifically, denoting a mention as m, and a cluster as C, we learn a function $f(m_i, C_y)$ that predicts the likelihood of an appropriate, positive merging of (C_x, C_y), where $m_i \in C_x$ and $C_x \neq C_y$.

Let $d(m_i, m_j)$ be the mention-pair distance score emitted by our CCNN model, while $m_i \in C_x$, and $m_j \in C_y$. Function $f(m_i, C_y)$ is based on four simple features:

- min-pair distance: $d(m_i, m_j)$
- avg-pair distance: $\frac{\sum_{m_j \in C_y} d(m_i, m_j)}{\|C_y\|}$
- max-pair distance: $d(m_i, m_j)$
- size of candidate cluster: $\|C_y\|$

The first three features serve to better represent the cluster at large. For example, if a given m_i were evaluated against two other clusters C_1 and C_2, it might be the case that both clusters of have the same min-pair distance score. Yet, the *distribution* of distance scores with the other mentions in each cluster might shed light onto which cluster has more similar mentions. We experimented with including the full distribution of distance scores, along with the variance in distance scores, but we received the best results from the min, avg, max distances – probably because most golden clusters contain just 1-4 mentions. The last feature (cluster size) serves is an explicit measure to help prevent clusters from growing too large.

At testing time, we evaluate every (m_i, C_y) pair.

We define f as a feed-forward neural network with 1 hidden layer of 25 units. We use ReLU activation without dropout, and our model predicts the probability of a positive merge via a 2-class softmax function. Our loss function is weighted binary cross-entropy, to account for the class imbalance situation that most mentions should not be merged with most clusters. We optimize with Adagrad. Like agglomerative, at each iteration, we merge only the (m_i, C_y) pair that yielded the highest score (likelihood of a positive merge). We continue to merge until model no longer predicts a merge (when all candidate pairs are < 0.5). Thus, unlike the aforementioned models, we do not require an additional parameter for our stopping criterion.

Our pseudocode is shown in Algorithm 1.

Algorithm 1 Neural Clustering

Require: $n \geq 0$

Note, since our neural classifier requires as training data the mention-pair scores predicted by CCNN, our model is limited to using the dev set's scores as training data. We also considered (1) adjusting the train/dev set sizes, yielding more dev data on which our clustering model could train; and (2) cross-validation. Yet, the original train/dev/test split performed the best, signifying that it is most important for the CCNN model to have as much training data as possible.

6 Results

We evaluated against the gold test mentions which were identified in ECB+.

Since our model requires representing each mention independently of the other candidate mention, our features could not contain relational-type information (e.g., SameLemma, # of sentences between Mentions, etc)[3]. We experimented with the 5 features detailed in Section 4.3: POS Embeddings, Lemma Embeddings, Dependency Lemma Embeddings, Character Embeddings, Word Embeddings.

Most coreference systems include dozens of features; however, in the interest of understanding which features were most useful, we built-up our system in an additive manner: we started with just one feature, testing each feature independently. If features seemed to help, we considered combining them with others. The results from all feature combinations are illustrated in Table **??**.

Lemma Embeddings were the most useful feature, followed closely by Character Embeddings. Since *SameLemma* historically has been a strong, difficult baseline, this is unsurprising. Using the embeddings of the lemmas, especially with the power of the CCNN, provides much more expression than the mere boolean feature of *SameLemma*. Our best results for this feature came from using the pre-trained 840 billion-token GloVe embeddings.

Character Embeddings were effective for the same reason String Edit Distance is often a strong feature. It is particularly useful for our corpus because the gold mentions for a given cluster are usually textually very similar to one another, if not identical. 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.

Combining Lemma Embeddings with Character Embeddings yielded **our highest performance – 8X.X CoNLL F1 score.** As this is the first event coreference system to test on the ECB+ golden test mentions, we denote this score as the new baseline to which to compare future systems.

It is clear that Lemma + Character Embeddings are complementary; the semantic information conveyed within the lemma embeddings complement the syntactic information of character embeddings. Related, POS by itself was a poor feature. However, combining POS with either Lemma Embeddings or Character Embeddings offered strong results. Interestingly, the Iteratively-Unfolding model used Lemma + POS (2017).

[3]Although, we did experiment with extending our CCNN model by adding relational features as a merged-layer at the highest neural level. However, doing so provided no benefit.

Ideally, a classifier should precisely learn how to combine all features smartly, such that the unhelpful features are given 0 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 to expand appropriately.

6.1 Comparison to Other Systems

HDDCRP chose to not use the gold test mentions provided by ECB+. Instead, they used a semantic role labeller to predict and filter all mentions. To compare against their system, we spent an enormous effort working with the HDDCRP source code and data to ensure we accurately use their exact predicted mentions. Of the 3,290 gold mentions in ECB+, their system failed to find/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. We outperform their system on this exact test set, as shown in Table **??**.

Iteratively-Unfolding attempted to use the same predicted test mentions as HDDCRP. As mentioned in Section 3.1, they removed false positives and singleton-mentions which belong to events that are only found within the given document (no cross-doc references). After numerous correspondences, we compare our system against the same set of test mentions they used, and the results are also shown in Table **??**.

Conclusion

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, CorefApp '99, pages 1–8. http://dl.acm.org/citation.cfm?id=1608810.1608812.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 1412–1422. http://dl.acm.org/citation.cfm?id=1858681.1858824.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '08, pages 294–303. http://dl.acm.org/citation.cfm?id=1613715.1613756.

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

Prafulla Kumar Choubey and Ruihong Huang. 2017. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *EMNLP*. http://www.aclweb.org/anthology/D17-1225.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. *CoRR* abs/1609.08667. http://cs.stanford.edu/people/kevclark/resources/clark-manning-emnlp2016-deep.pdf.

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. http://arxiv.org/abs/1606.01323.

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). http://www.aclweb.org/anthology/L14-1646.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT-NAACL-DUC '03, pages 9–16. https://doi.org/10.3115/1119467.1119469.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*. ACL, pages 1971–1982. http://nlp.cs.berkeley.edu/pubs/Durrett-Klein_2013_Coreference_paper.pdf.

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

Richard Zemel Ruslan Salakhutdinov Gregory Koch. 2015. Siamese neural networks for one-shot image recognition. http://www.cs.toronto.edu/~rsalakhu/papers/oneshot1.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, ANARESOLUTION '97, pages 75–81. http://dl.acm.org/citation.cfm?id=1598819.1598830.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. ACL, pages 1746–1751. http://aclweb.org/anthology/D/D14/D14-1181.pdf.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, New York, New York, USA, volume 48 of *Proceedings of Machine Learning Research*, pages 507–516. http://proceedings.mlr.press/v48/liud16.html.

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. http://www.aclweb.org/anthology/P/P14/P14-5010.

James Mayfield and et al. 2009. Cross-Document Coreference Resolution: A Key Technology for Learning by Reading. In *Proceedings of the AAAI 2009 Spring Symposium on Learning by Reading and Learning to Read*. AAAI Press. http://ebiquity.umbc.edu/_file_directory_/papers/442.pdf.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '04. https://doi.org/10.3115/1220355.1220455.

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. http://www.aclweb.org/anthology/D14-1162.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*. IEEE Computer Society, pages 815–823. http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#SchroffKP15.

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. http://dl.acm.org/citation.cfm?id=972597.972602.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016a. Learning global features for coreference resolution. *CoRR* abs/1604.03035. http://nlp.seas.harvard.edu/papers/corefmain.pdf.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016b. Learning global features for coreference resolution. In *HLT-NAACL*. The Association for Computational Linguistics, pages 994–1004. http://aclweb.org/anthology/N/N16/N16-1114.pdf.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL (1)*. The Association for Computer Linguistics, pages 1416–1426. http://aclweb.org/anthology/P/P15/P15-1137.pdf.

Bishan Yang, Claire Cardie, and Peter I. Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *TACL* 3:517–528. http://dblp.uni-trier.de/db/journals/tacl/tacl3.html#YangCF15.

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.

A Supplemental Material

Submissions may include resources (software and/or data) used in in the work and described in the paper. Papers that are submitted with accompanying software and/or data may receive additional credit toward the overall evaluation score, and the potential impact of the software and data will be taken into account when making the acceptance/rejection decisions. Any accompanying software and/or data should include licenses and documentation of research review as appropriate.

NAACL-HLT 2018 also encourages the submission of supplementary material to report preprocessing decisions, model parameters, and other details necessary for the replication of the experiments reported in the paper. Seemingly small preprocessing decisions can sometimes make a large difference in performance, so it is crucial to record such decisions to precisely characterize state-of-the-art methods.

Nonetheless, supplementary material should be supplementary (rather than central) to the paper. **Submissions that misuse the supplementary material may be rejected without review.** Essentially, supplementary material may include explanations or details of proofs or derivations that do not fit into the paper, lists of features or feature templates, sample inputs and outputs for a system, pseudo-code or source code, and data. (Source code and data should be separate uploads, rather than part of the paper).

The paper should not rely on the supplementary material: while the paper may refer to and cite the supplementary material and the supplementary material will be available to the reviewers, they will not be asked to review the supplementary material.

Appendices (*i.e.* supplementary material in the form of proofs, tables, or pseudo-code) should come after the references, as shown here. Use `\appendix` before any appendix section to switch the section numbering over to letters.

B Multiple Appendices

. . . can be gotten by using more than one section. We hope you won't need that.