Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers

Nafise Sadat Moosavi[1,2*] and **Michael Strube**[1]
[1]Heidelberg Institute for Theoretical Studies gGmbH
[2]Research Training Group AIPHES
moosavi@ukp.informatik.tu-darmstadt.de, michael.strube@h-its.org

Abstract

Coreference resolution is an intermediate step for text understanding. It is used in tasks and domains for which we do not necessarily have coreference annotated corpora. Therefore, generalization is of special importance for coreference resolution. However, while recent coreference resolvers have notable improvements on the CoNLL dataset, they struggle to generalize properly to new domains or datasets. In this paper, we investigate the role of linguistic features in building more generalizable coreference resolvers. We show that generalization improves only slightly by merely using a set of additional linguistic features. However, employing features and subsets of their values that are informative for coreference resolution, considerably improves generalization. Thanks to better generalization, our system achieves state-of-the-art results in out-of-domain evaluations, e.g., on WikiCoref, our system, which is trained on CoNLL, achieves on-par performance with a system designed for this dataset.

1 Introduction

Coreference resolution is the task of recognizing different expressions that refer to the same entity. The referring expressions are called mentions. For instance, the sentence "[Susan]$_1$ sent [her]$_1$ daughter to a boarding school" contains two coreferring mentions. "her" is an anaphor which refers to the antecedent "Susan".

The availability of coreference information benefits various Natural Language Processing (NLP) tasks including automatic summarization, question answering, machine translation and information extraction. Current coreference developments are almost only targeted at improving scores on the CoNLL official test set. However, the superiority of a coreference resolver on the CoNLL evaluation sets does not necessarily indicate that it also performs better on new datasets. For instance, the ranking model of Clark and Manning (2016a), the reinforcement learning model of Clark and Manning (2016b) and the end-to-end model of Lee et al. (2017) are three recent coreference resolvers, among which the model of Lee et al. (2017) performs the best and that of Clark and Manning (2016b) performs the second best on the CoNLL development and test sets. However, if we evaluate these systems on the WikiCoref dataset (Ghaddar and Langlais, 2016a), which is consistent with CoNLL with regard to coreference definition and annotation scheme, the performance ranking would be in a reverse order[1].

In Moosavi and Strube (2017a), we investigate the generalization problem in coreference resolution and show that there is a large overlap between the coreferring mentions in the CoNLL training and evaluation sets. Therefore, higher scores on the CoNLL evaluation sets do not necessarily indicate a better coreference model. They may be due to better memorization of the training data. As a result, despite the remarkable improvements in coreference resolution, the use of coreference resolution in other applications is mainly limited to the use of simple rule-based systems, e.g. Lapata and Barzilay (2005),Yu and Ji (2016), and Elsner and Charniak (2008).

In this paper, we explore the role of linguistic features for improving generalization. The incorporation of linguistic features is considered as a potential solution for building more generalizable NLP systems[2]. While linguistic features[3]

[*] This author is currently employed by the Ubiquitous Knowledge Processing (UKP) Lab, Technische Universität Darmstadt, https://www.ukp.tu-darmstadt.de.

[1] The single model of Lee et al. (2017) is used here.

[2] E.g. there is a dedicated workshop for this topic https://sites.google.com/view/relsnnlp.

[3] We refer to features that are based on linguistic intu-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 193–203
Brussels, Belgium, October 31 - November 4, 2018.

were shown to be important for coreference resolution, e.g. Uryupina (2007) and Bengtson and Roth (2008), state-of-the-art systems no longer use them and mainly rely on word embeddings and deep neural networks. Since all recent systems are using neural networks, we focus on the effect of linguistic features on a neural coreference resolver.

The contributions of this paper are as follows:

- We show that linguistic features are more beneficial for a neural coreference resolver if we incorporate features and subsets of their values that are informative for discriminating coreference relations. Otherwise, employing linguistic features with all their values only slightly affects the performance and generalization.
- We propose an efficient discriminative pattern mining algorithm, called EPM, for determining (feature, value) pairs that are informative for the given task. We show that while the informativeness of EPM mined patterns is on-par with those of its counterparts, it scales best to large datasets.[4]
- By improving generalization, we achieve state-of-the-art performance on all examined out-of-domain evaluations. Our out-of-domain performance on WikiCoref is on-par with that of Ghaddar and Langlais (2016b)'s coreference resolver, which is a system specifically designed for WikiCoref and uses its domain knowledge.

2 Importance of Features in Coreference

Uryupina (2007)'s thesis is one of the most thorough analyses of linguistically motivated features for coreference resolution. She examines a large set of linguistic features, i.e. string match, syntactic knowledge, semantic compatibility, discourse structure and salience, and investigates their interaction with coreference relations. She shows that even imperfect linguistic features, which are extracted using error-prone preprocessing modules, boost the performance and argues that coreference resolvers could and should benefit from linguistic theories. Her claims are based on analyses on the MUC dataset. Ng and Cardie (2002), Yang et al. (2004), Ponzetto and Strube (2006), Bengtson and Roth (2008), and Recasens and Hovy (2009) also study the importance of features in coreference resolution.

Apart from the mentioned studies, which are mainly about the importance of individual features, studies like Björkelund and Farkas (2012), Fernandes et al. (2012), and Uryupina and Moschitti (2015) generate new features by combining basic features. Björkelund and Farkas (2012) do not use a systematic approach for combining features. Fernandes et al. (2012) use the Entropy guided Feature Induction (EFI) approach (Fernandes and Milidiú, 2012) to automatically generate discriminative feature combinations. The first step is to train a decision tree on a dataset in which each sample consists of features describing a mention pair. The EFI approach traverses the tree from the root in a depth-first order and recursively builds feature combinations. Each pattern that is generated by EFI starts from the root node. As a result, EFI tends to generate long patterns. A decision tree does not represent all patterns of data. Therefore, it is not possible to explore all feature combinations from a decision tree.

Uryupina and Moschitti (2015) propose an alternative approach to EFI. They formulate the problem of generating feature combinations as a pattern mining approach. They use the Jaccard Item Mining (JIM) algorithm[5] (Segond and Borgelt, 2011). They show that the classifier that uses the JIM features significantly outperforms the one that employs the EFI features.

3 Baseline Coreference Resolver

deep-coref (Clark and Manning, 2016a) and e2e-coref (Lee et al., 2017) are among the best performing coreference resolvers from which e2e-coref performs better on the CoNLL test set. deep-coref is a pipelined system, i.e. a mention detection first determines the list of candidate mentions with their corresponding features. It contains various coreference models including the mention-pair, mention-ranking, and entity-based models. The mention-ranking model of deep-coref has three variations: (1) "ranking" uses the slack-rescaled max-margin training objective of Wiseman et al. (2015), (2) "reinforce" is a variation of the "ranking" model in which the hyper-parameters are set in a reinforcement learning framework (Sutton and Barto, 1998), and (3) "top-

itions, e.g. string match, or are acquired from linguistic preprocessing modules, e.g. POS tags, as linguistic features.

[4] The EPM code is available at https://github.com/ns-moosavi/epm

[5] http://www.borgelt.net/jim.html

pairs" is a simple variation of the "ranking" model that uses a probabilistic objective function and is used for pretraining the "ranking" model.

e2e-coref is an end-to-end system that jointly models mention detection and coreference resolution. It considers all possible (start, end) word spans of each sentence as candidate mentions. Apart from a single model, e2e-coref includes an ensemble of five models.

We use deep-coref as the baseline in our experiments. The reason is that some of the examined features require the head of each mention to be known, e.g. head match, while e2e-coref mentions do not have specific heads and heads are automatically determined using an attention mechanism. We also observe that if we limit e2e-coref candidate spans to those that correspond to deep-coref's detected mentions, the performance of e2e-coref drops to a level on-par with deep-coref[6].

4 Examined Features

The examined linguistic features include string match, syntactic, shallow semantic and discourse features. **Mention-based** features include:

- Mention type: proper, nominal or pronominal
- Fine mention type: proper, definite or indefinite nominal, or the citation form of pronouns
- Gender: female, male, neutral, unknown
- Number: singular, plural, unknown
- Animacy: animate, inanimate, unknown
- Named entity type: person, location, organization, date, time, number, etc.
- Dependency relation: enhanced dependency relation (Schuster and Manning, 2016) of the head word to its parent
- POS tags of the first, last, head, two words preceding and following of each mention

Pairwise features include:

- Head match: both mentions have the same head, e.g. "red hat" and "the hat"
- String of one mention is contained in the other, e.g. "Mary's hat" and "Mary"
- Head of one mention is contained in the other, e.g. "Mary's hat" and "hat"
- Acronym, e.g. "Heidelberg Institute for Theoretical Studies" and "HITS"
- Compatible pre-modifiers: the set of pre-modifiers of one mention is contained in that of the other, e.g. "the red hat that she is wearing" and "the red hat"
- Compatible[7] gender, e.g. "Mary" and "women"
- Compatible number, e.g. "Mary" and "John"
- Compatible animacy, e.g. "those hats" and "it"
- Compatible attributes: compatible gender, number and animacy, e.g. "Mary" and "she"
- Closest antecedent that has the same head and compatible premodifiers, e.g. "this new book" and "This book" in "Take a look at this new book. This book is one of the best sellers."
- Closest antecedent that has compatible attributes, e.g. the antecedent "Mary" and the anaphor "she" in the sentence "John saw Mary, and she was in a hurry"
- Closest antecedent that has compatible attributes and is a subject, e.g. the antecedent "Mary" and the anaphor "she" in the sentence "Mary saw John, but she was in a hurry"
- Closest antecedent that has compatible attributes and is an object, e.g. "Mary" and "she" in "John saw Mary, and she was in a hurry"

The last three features are similar to the discourse-level features discussed by Uryupina (2007), which are created by combining *proximity*, *agreement* and *salience* properties. She shows that such features are useful for resolving pronouns. we estimate proximity by considering the distance of two mentions. The salience is also incorporated by discriminating subject or object antecedents. We do not use any gold information. All features are extracted using Stanford CoreNLP (Manning et al., 2014).

5 Impact of Linguistic Features

In this section, we examine the effect of employing all linguistic features described in Section 4 in a neural coreference resolver, i.e. deep-coref. We use *MUC* (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), $CEAF_e$ (Luo, 2005), *LEA* (Moosavi and Strube, 2016), and the *CoNLL* score (Pradhan et al., 2014), i.e. the average F_1 value of *MUC*, B^3, and $CEAF_e$, for evaluations.

The results of employing those features in deep-coref's "ranking" and "top-pairs" models on the

[6] The CoNLL score of the e2e-coref single model on the CoNLL development set drops from 67.36 to 65.81, while that of the deep-coref "ranking" model is 66.09.

[7] One value is unknown, or both values are identical.

CoNLL development set are reported in Table 1.

	MUC	B^3	CEAF$_e$	CoNLL	LEA
ranking	74.31	64.23	59.73	66.09	60.47
+linguistic	74.35	63.96	60.19	66.17	60.20
top-pairs	73.95	63.98	59.52	65.82	60.07
+linguistic	74.32	64.45	60.19	66.32	60.62

Table 1: Impact of linguistic features on deep-coref models on the CoNLL development set.

The rows "ranking" and "top-pairs" show the base results of deep-coref's "ranking" and "top-pairs" models, respectively. "+linguistic" rows represents the results for each of the mention-ranking models in which the feature set of Section 4 is employed. The gender, number, animacy and mention type features, which have less than five values, are converted to binary features. Named entity and POS tags, and dependency relations are represented as learned embeddings.

We observe that incorporating all the linguistic features bridges the gap between the performance of "top-pairs" and "ranking". However, it does not improve significantly over "ranking". Henceforth, we use the "top-pairs" model of deep-coref as the baseline model to incorporate linguistic features.

To assess the impact on generalization, we evaluate "top-pairs" and "+linguistic"[8] models that are trained on CoNLL, on WikiCoref (see Table 2). We observe that the impact on generalization is also not notable, i.e. the CoNLL score improves only by 0.5pp over "ranking".

	MUC	B^3	CEAF$_e$	CoNLL	LEA
ranking	63.10	48.43	47.18	52.90	44.40
top-pairs	63.09	48.42	46.05	52.52	44.21
+linguistic	63.99	49.63	46.60	53.40	45.66

Table 2: Out-of-domain evaluation of deep-coref models on the WikiCoref dataset.

Based on an ablation study, while our feature set contains numerous features, the resulting improvements of "linguistic" over "top-pairs" mainly comes from the last four pairwise features in Section 4, which are carefully designed features.

6 Better Exploiting Linguistic Features

As discussed by Moosavi and Strube (2017a), there is a large lexical overlap between the coreferring mentions of the CoNLL training and evaluation sets. As a result, lexical features provide a very strong signal for resolving coreference relations.

For linguistic features to be more effective in current coreference resolvers, which rely heavily on lexical features, they should also provide a strong signal for coreference resolution.

Additional linguistic features are not necessarily all informative for coreference resolution, especially if they are extracted automatically and are noisy. Besides, for features with multiple values, e.g. mention-based features, only a small subset of values may be informative.

To better exploit linguistic features, we only employ (feature, value) pairs[9] that are informative for coreference resolution. Coreference resolution is a complex task in which features have complex interactions (Recasens and Hovy, 2009). As a result, we cannot determine the informativeness of feature-values in isolation.

We use a discriminative pattern mining approach (Cheng et al., 2007, 2008; Batal and Hauskrecht, 2010) that examines all combinations of feature-values, up to a certain length, and determines which feature-values are informative when they are considered in combination.

Due to the large data size (all mention-pairs of the CoNLL training data) and the high dimensionality of feature-values, compared to common evaluation sets of pattern mining methods, the existing discriminative pattern mining approaches were not applicable to our data. In this section, we propose an efficient discriminative pattern mining approach, called Efficient Pattern Miner (EPM), that is scalable to large NLP datasets. The most important properties of EPM are (1) it examines all frequent feature-values combinations, up to the desired length, (2) it is scalable to large datasets, and (3) it is only data dependent and independent of the coreference resolver.

6.1 Notation

We use the following notations and definitions throughout this section:

- $D = \{X_i, c(X_i)\}_{i=1}^{n}$: set of n training samples. X_i is the set of feature-values that describes the ith sample. $c(X_i) \in C$ is the label of X_i, e.g. coreferent and non-coreferent.
- $A = \{a_1, \ldots, a_l\}$: set of all feature-values present in D. Each $a_i \in A$ is called an item, e.g. $a_i =$"anaphor type=proper".

[8] i.e. "top-pairs+linguistic"

[9] Henceforth, we refer to them as feature-values.

- p: pattern $p = \{a_{i_1}, \ldots, a_{i_k}\}$ is a set of one or more items, e.g. p ={"anaphor type=proper", "antecedent type=proper"}.
- $support(p, c_i)$: the number of samples that contain pattern p and are labeled with c_i.

6.2 Data Structure

For representing the input samples, we use the Frequent Pattern Tree (FP-Tree) structure that is the data structure of the FP-Growth algorithm (Han et al., 2004), i.e. one of the most common algorithms for frequent pattern mining. FP-Tree provides a structure for representing all existing patterns of data in a compressed form. Using the FP-Tree structure allows an efficient enumeration of all frequent patterns. In the FP-Tree structure, items are arranged in descending order of frequency. Frequency of an item corresponds to $\sum_{c_i \in C} support(a_i, c_i)$. Except for the root, which is a null node, each node n contains an item $a_i \in A$. It also contains the support values of a_i in the subpath of the tree that starts from the root and ends with n, i.e. $support_n(a_i, c_j)$.

The FP-Tree construction method (Han et al., 2004) is as follows: (a) scan D to collect the set of all items, i.e. A. Compute $support(a_i, c_j)$ for each item $a_i \in A$ and label $c_j \in C$. Sort A's members in descending order according to their frequencies, i.e. $\sum_{c_i \in C} support(a_i, c_i)$. (b) create a null-labeled node as the root, and (c) scan D again. For each $(X_i, c(X_i)) \in D$:

1. Order all items $a_j \in X_i$ according to the order in A.
2. Set the current node (T) to the root.
3. Consider $X_i = [a_k|\bar{X}_i]$, where a_k is the first (ordered) item of x_i, and $\bar{X}_i = X_i - a_k$. If T has a child n that contains a_k then increment $support_n(a_k, c(X_i))$ by one. Otherwise, create a new node n that contains a_k with $support_n(a_k, c(X_i)) = 1$. Add n to the tree as a child of T.
4. If $\bar{X}_i$ is non-empty, set T to n. Assign $X_i = \bar{X}_i$ and go to step 3.

As an example, assume D contains the following two samples:

X_1={ana-type=NAM, ant-type=NAM, head-match=F}, $C(X_1) = 0$

X_2={ana-type=NAM, ant-type=NAM, head-match=T}, $C(X_2) = 1$

Based on these samples A={ana-type=NAM, ant-type=NAM, head-match=F, head-match=T}, $support(a_i, 0)_{a_i \in A}$= {1,1,1,0}, and $support(a_i, 1)_{a_i \in A}$={1,1,0,1}. If we sort A based on a_i's frequencies ($support(a_i, 0) + support(a_i, 1)$), the ordering of A's items will remain the same.

The FP-Tree construction steps for the above samples are demonstrated in Figure 1. ana-type, ant-type, and head-match features are abbreviated as ana, ant, and head, respectively.

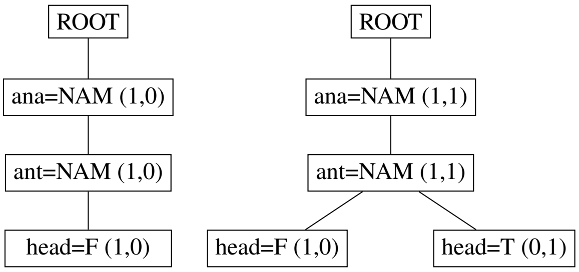

Figure 1: Left to right: (partially) constructed FP-Tree for the example in Section 6.2.

From an initial FP-Tree (T) that represents all existing patterns, one can easily obtain a new FP-Tree in which all patterns include a given pattern p. This can be done by only including sub-paths of T that contain pattern p. The new tree is called conditional FP-Tree of p, T_p. An example of conditional FP-Tree is included in the supplementary materials.

6.3 Informativeness Measures

We use a *discriminative power* and an *information novelty* measure for determining informativeness. We also use a *frequency* measure which determines the required minimum frequency of a pattern in training samples. It helps to avoid overfitting to the properties of the training data.

Discriminative power: We use the G^2 likelihood ratio test (Agresti, 2007) in order to choose patterns whose association with the class variable is statistically significant.[10] The G^2 test is successfully used for text analysis (Dunning, 1993).

Information Novelty: A large number of redundant patterns can be generated by adding irrelevant items to a base pattern that is discriminative itself.

[10] A pattern is considered discriminative if the corresponding p-value is less than a fixed threshold (0.01).

We consider the pattern p as novel if (1) p predicts the target class label c significantly better than all of its containing items, and (2) p predicts c significantly better than all of its sub-patterns that satisfy the frequency, discriminative power, and the first information novelty conditions. Similar to Batal and Hauskrecht (2010), we employ a binomial distribution to determine information novelty.

6.4 Mining Algorithm

The EPM algorithm is summarized in Algorithm 1. It takes FP-Tree T, pattern p on which T is conditioned, and set of items ($A_j \subset A$) whose combinations with p will be examined. Initially, p is empty and the FP-Tree is constructed based on all frequent items of data and $A_j = A$. Resulting patterns are collected in P.

For each $a_i \in A_j$, the algorithm builds new pattern q by combining a_i with p. $frequent(q)$ checks whether q meets the frequency condition. If q is frequent, the algorithm continues the search process. Otherwise, it is not possible to build any frequent pattern out of a non-frequent one. *Discriminative power* and the first condition of *information novelty* are then checked for pattern q.

```
Algorithm $EPM$ ($T$, $p$, $A_j$)
    foreach $a_i \in A_j$ do
        $q = p \cup \{a_i\}$
        if $Frequent(q)$ then
            if $Discriminative(q)$ then
                if $Novel(q)$ then
                    $P = P \cup q$
                end
            end
            if $|q| >= \Theta_l$ then
                continue
            end
            construct $T_q$ = $q$'s conditional tree
            $EPM(T_q, q, ancestors(a_i))$
        end
    end
```

Algorithm 1: The EPM algorithm.

We use a threshold (Θ_l) for the maximum length of mined patterns. Θ_l can be set to large values if more complex and specific patterns are desirable.

If $|q|$ is smaller than Θ_l, the conditional FP-Tree T_q is built that represents patterns of T that include the pattern q. The mining algorithm then continues to recursively search for more specific patterns by combining q with the items included in $ancestors(a_i)$, which keeps the list of all ancestors of a_i in the original FP-Tree. EPM examines all frequent patterns of up to length Θ_l.

If we use a statistical test multiple times, the risk of making false discoveries increases (Webb, 2006). To tackle this, we apply the Bonferroni correction for multiple tests in a post-pruning function after the mining process. This function also applies the second information novelty condition on the resulting patterns.

7 Why Use EPM?

In this section, we explain why EPM is a better alternative compared to its counterparts for large NLP datasets. We compare EPM with two efficient discriminative pattern mining algorithms, i.e. Minimal Predictive Patterns (MPP) (Batal and Hauskrecht, 2010) and Direct Discriminative Pattern Mining (DDPMine) (Cheng et al., 2008), on standard machine learning datasets.

MPP selects patterns that are significantly more predictive than all their sub-patterns. It measures significance by the binomial distribution. For each pattern of length l, MPP checks $2^l - 1$ sub-patterns. DDPMine is an iterative approach that selects the most discriminative pattern at each iteration and reduces the search space of the next iteration by removing all samples that include the selected pattern. DDPMine uses the FP-Tree structure.

We show that EPM scales best and compares favorably based on the informativeness of resulting patterns. Due to its efficiency, EPM can handle large datasets similar to ones that are commonly used in various NLP tasks.

7.1 Experimental Setup

We use the same FP-Tree implementation for DDPMine and EPM. In all algorithms, we consider a pattern as frequent if it occurs in 10% of the samples of one of the classes. We use $\Theta_l = 3$ for both MPP and EPM.

We perform 5-times repeated 5-fold cross validation and the results are averaged. In each validation, all experiments are performed on the same split. We use a linear SVM, i.e. LIBLINEAR 2.11 (Fan et al., 2008), as the baseline classifier.

We use several datasets from the UCI machine learning repository (Lichman, 2013) whose characteristics are presented in the first three columns of Table 3, i.e. the number of (1)

	Data characteristics			# Patterns			Micro-F				Macro-F			
Dataset	#Features	#FI	n	DDP	MPP	EPM	Orig	DDP	MPP	EPM	Orig	DDP	MPP	EPM
cmc	(0/2/7)	24	1473	4	99	23	77.5	77.4	76.2	77.3	57.3	57.1	57.7	59.4
nursery	(0/0/8)	27	12690	4	258	198	97.5	98.2	99.9	99.8	49.4	79.4	99.8	98.8
sick	(6/1/22)	36	2800	5	627	89	94.6	94.7	96.1	95.8	62.6	64.8	81.0	75.6
kr-v-k	(0/0/16)	40	28056	7	71	63	99.1	99.1	99.6	99.6	49.8	49.8	87.8	88.4
german	(0/7/13)	51	1000	8	548	97	70.7	70.9	73.1	72.7	49.6	55.2	65.3	64.2
connect-4	(0/0/42)	75	67557	-	-	907	90.5	-	-	90.5	47.5	-	-	56.6
census	(1/12/28)	76	299284	-	-	5618	93.8	-	-	93.8	48.4	-	-	51.6
poker	(0/10/0)	85	1025010	-	-	14216	23.1	-	-	49.6	22.4	-	-	44.5

Table 3: Evaluating the informativeness of DDPMine, MPP and EPM patterns on standard datasets.

(real/integer/nominal) features (#Features), (2) frequent items (#FI), and (3) samples (n). We use one[the minority class]-vs-all technique for datasets with more than two classes.

7.2 How Informative are EPM Patterns?

To evaluate the informativeness of mined patterns, the common practice is to add them as new features to the feature set of the baseline classifier; the more informative the patterns, the greater impact they would have on the overall performance. All patterns are added as binary features, i.e. the feature is true for samples that contain all items of the corresponding pattern.

The effect of the patterns of DDPMine, MPP and EPM on the overall accuracy is presented in Table 3. The columns #Patterns show the number of patterns mined by each of the algorithms. The *Orig* columns show the results of the SVM using the original feature sets. The *DDP*, *MPP*, and *EPM* columns show the results of the SVM on the datasets for which the feature set is extended by the features mined by DDPMine, MPP, and EPM, respectively. The results of the 5-repeated 5-fold cross validation are reported if each single validation takes less than 10 hours.

Based on the results of Table 3 (1) EPM efficiently scales to larger datasets, (2) MPP and EPM patterns considerably improves the performance, and (3) EPM has on-par results with MPP while it mines considerably fewer patterns.

7.3 How Does it Scale?

Figure 2 compares EPM mining time (in seconds) with those of DDPMine and MPP. The parameter in the parentheses is the pattern size threshold, e.g. $\Theta_l = 4$ for EPM(4). The experiments that take more than two days are terminated and are not included. EPM is notably faster in comparison to the other two approaches. It is notable that the examined datasets are considerably smaller than the coreference data, which includes more than 33 million samples and 200 frequent feature-values.

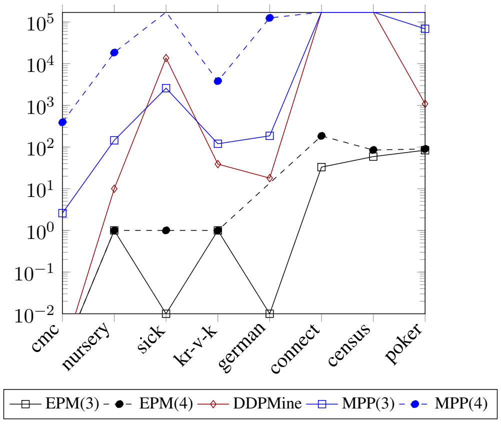

Figure 2: Comparison of mining times (seconds).

8 Impact of Informative Feature-values

8.1 Experimental Setup

For determining informative feature-values, we extract all features for all mention-pairs[11] of the CoNLL training data and then apply EPM on this data. In order to prevent learning annotation errors and specific properties of the training data, we consider a pattern as frequent if it occurs in coreference relations of at least m different coreferring anaphors ($m = 20$). Since the majority of mention-pairs are non-coreferent and we are not interested in patterns for non-coreferring relations, we also consider the coreference probability of each pattern p, i.e. $\frac{|\{X_i|p \in X_i \wedge c(X_i)=coreferent\}|}{|\{X_i|p \in X_i\}|}$, in the post-pruning function. The coreference probability should be higher than a threshold (60% in our experiments), so we only mine patterns that are informative for coreferring mentions.

For the coreference resolution experiments, instead of incorporating informative patterns, we incorporate feature-values that are included in the

[11] Each mention is paired with all the preceding mentions.

		MUC			B^3			CEAF$_e$			CoNLL	LEA		
		R	P	F$_1$	R	P	F$_1$	R	P	F$_1$		R	P	F$_1$
deep-coref	ranking	70.43	79.57	74.72	58.08	69.26	63.18	54.43	64.17	58.90	65.60	54.55	65.68	59.60
	reinforce	69.84	79.79	74.48	57.41	70.96	63.47	55.63	63.83	59.45	65.80	53.78	67.23	59.76
	top-pairs	69.41	79.90	74.29	57.01	70.80	63.16	54.43	63.74	58.72	65.39	53.31	67.09	59.41
	+EPM	71.16	79.35	75.03	59.28	69.70	64.07	56.52	64.02	60.04	66.38	55.63	66.11	60.42
	+JIM	69.89	80.45	74.80	57.08	71.58	63.51	55.36	64.20	59.45	65.93	53.46	67.97	59.85
e2e	single	74.02	77.82	75.88	62.58	67.45	64.92	59.16	62.96	61.00	67.27	58.90	63.79	61.25
	ensemble	73.73	80.95	77.17	61.83	72.10	66.57	60.11	65.62	62.74	68.83	58.48	68.81	63.23

Table 4: Comparisons on the CoNLL test set. The F$_1$ gains that are statistically significant: (1) "+EPM" compared to "top-pairs", "ranking" and "JIM", (2) "+EPM" compared to "reinforce" based on MUC, B^3 and LEA, (3) "single" compared to "+EPM" based on MUC and B^3, and (4) "ensemble" compared to other systems. Significance is measured based on the approximate randomization test ($p < 0.05$) (Noreen, 1989).

informative patterns mined by EPM. The reason is that deep-coref, or any other recent coreference resolver, uses a deep neural network, which has a fully automated feature generation process. We add these feature-values as binary features.

By setting Θ_l to five,[12] EPM results in 13 pairwise feature-values, 112 POS tags, i.e. 53 POS for anaphors and 59 for antecedents, 25 dependency relations, 26 mention types (mention types or fine mention types), and finally, 14 named entity tags.[13]

Based on the observation in Section 5, we use the top-pairs model of deep-coref as the baseline to employ additional features, i.e. "+EPM" is the top-pairs model in which EPM feature-values are incorporated.

8.2 Impact on In-domain Performance

The performance of the "+EPM" model compared to recent state-of-the-art coreference models on the CoNLL test set is presented in Table 4. The "single" and "ensemble" rows represent the results of the single and ensemble models of e2e-coref.

We also compare EPM with the pattern mining approach used by Uryupina and Moschitti (2015), i.e. Jaccard Item Mining (JIM). For a fair comparison, while Uryupina and Moschitti (2015) used mined patterns for extracting feature templates, we use them for selecting feature-values. We run the JIM algorithm on the same data and with the same setup as that of EPM.[14] This results in nine pairwise features, 260 POS tags, 38 dependency relations, 32 mention types, and 18 named entity tags. The "+JIM" row shows the results of deep-coref top-pairs model in which these feature-values are incorporated. As we see, EPM feature-values result in significantly better performance than those of JIM while the number of EPM feature-values is considerably less than JIM.

	MUC	B^3	CEAF$_e$	CoNLL	LEA
+EPM	74.92	65.03	60.88	66.95	61.34
-pairwise	74.37	64.55	60.46	66.46	60.71
-type	74.71	64.87	61.00	66.86	61.07
-dep	74.57	64.79	60.65	66.67	61.01
-NER	74.61	65.05	60.93	66.86	61.27
-POS	74.74	65.04	60.88	66.89	61.30
+pairwise	74.25	64.33	60.02	66.20	60.57

Table 5: Impact of different EPM feature groups on the CoNLL development set.

Feature Ablation Table 5 shows the effect of each group of EPM feature-values, i.e. pairwise features, mention types, dependency relations, named entity tags and POS tags, on the performance of "+EPM". The performance of "+EPM" from which each of the above feature groups is removed, one feature group at a time, is represented as "-pairwise", "-types", "-dep", "-NER", and "-POS", respectively. The POS and named entity tags have the least and the pairwise features have the most significant effect. Since pairwise features have the most significant effect, we also perform an experiment in which only pairwise features are incorporated in the "top-pairs" model, i.e. "+pairwise". The results of "-pairwise" compared to "+pairwise" show that pairwise feature-values have a significant impact, but only when they are considered in combination with other EPM

[12] We observe that using larger Θ_l values will result in many over-specified patterns.

[13] Following the previous studies that show different features are of different importance for various types of mentions, e.g. Denis and Baldridge (2008) and Moosavi and Strube (2017b), we mine a separate set of patterns for each type of anaphor. These resulting feature-values are the union of informative feature-values for all types of anaphora.

[14] We set the minimum frequency, maximum pattern length and $score^+$ threshold parameters of JIM to 20, 5 and 0.6.

		MUC			B^3			CEAF$_e$			CoNLL	LEA		
		R	P	F$_1$	R	P	F$_1$	R	P	F$_1$		R	P	F$_1$
e2edeep-coref	ranking	57.72	69.57	63.10	41.42	58.30	48.43	42.20	53.50	47.18	52.90	37.57	54.27	44.40
	reinforce	62.12	58.98	60.51	46.98	45.79	46.38	44.28	46.35	45.29	50.73	42.28	41.70	41.98
	top-pairs	56.31	71.74	63.09	39.78	61.85	48.42	40.80	52.85	46.05	52.52	35.87	57.58	44.21
	+EPM	58.23	74.05	**65.20**	43.33	63.90	51.64	43.44	56.33	**49.05**	**55.30**	39.70	59.81	**47.72**
	single	60.14	64.46	62.22	45.20	51.75	48.25	38.18	43.50	40.67	50.38	40.70	47.56	43.86
	ensemble	59.58	71.60	65.04	44.64	60.91	51.52	40.38	49.17	44.35	53.63	40.73	56.97	47.50
	G&L	66.06	62.93	64.46	57.73	48.58	**52.76**	46.76	49.54	48.11	55.11	-	-	-

Table 6: Out-of-domain evaluation on the WikiCoref dataset. The highest F_1 scores are boldfaced.

feature-values.

		in-domain		out-of-domain	
		CoNLL	LEA	CoNLL	LEA
		pt (Bible)			
deep-coref	ranking	75.61	71.00	66.06	57.58
	+EPM	76.08	71.13	**68.14**	**60.74**
e2e-coref	single	77.80	73.73	65.22	58.26
	ensemble	**78.88**	**74.88**	65.45	59.71
		wb (weblog)			
deep-coref	ranking	61.46	53.75	57.17	48.74
	+EPM	61.97	53.93	**61.52**	**53.78**
e2e-coref	single	62.02	53.09	60.69	52.69
	ensemble	**64.76**	**57.54**	60.99	52.99

Table 7: In-domain and out-of-domain evaluations for the pt and wb genres of the CoNLL test set. The highest scores are boldfaced.

8.3 Impact on Generalization

We use the same setup as that of Moosavi and Strube (2017a) for evaluating generalization including (1) training on the CoNLL data and testing on WikiCoref[15] and (2) excluding a genre of the CoNLL data from training and development sets and testing on the excluded genre. Similar to Moosavi and Strube (2017a), we use the *pt* and *wb* genres for the latter evaluation setup.

The results of the first evaluation setup are shown in Table 6. The best performance on WikiCoref is achieved by Ghaddar and Langlais (2016a) ("G&L" in Table 6) who introduced WikiCoref and design a domain-specific coreference resolver that makes use of the Wikipedia markups of a document as well as links to Freebase, which are annotated in WikiCoref.

Incorporating EPM feature-values improves the performance by about three points. While "+EPM" does not use the WikiCoref data during training, and unlike "G&L", it does not employ any domain-specific features, it achieves on-par performance with that of "G&L". This indeed shows the effectiveness of informative feature-values in improving generalization.

The second set of generalization experiments is reported in Table 7. "in-domain" columns show the results when the evaluation genres were included in training and development sets while the "out-of-domain" columns show the results when the evaluation genres were excluded. As we can see, "+EPM" generalizes best, and in out-of-domain evaluations, it considerably outperforms the ensemble model of e2e-coref, which has the best performance on the CoNLL test set.

9 Conclusions

In this paper, we show that employing linguistic features in a neural coreference resolver significantly improves generalization. However, the incorporated features should be informative enough to be taken into account in the presence of lexical features, which are very strong features in the CoNLL dataset. We propose an efficient algorithm to determine informative feature-values in large datasets. As a result of a better generalization, we achieve state-of-the-art results in all examined out-of-domain evaluations.

Acknowledgments

The authors would like to thank Mark-Christoph Müller, Benjamin Heinzerling, Alex Judea, Steffen Eger and the anonymous reviewers for their helpful comments and feedbacks. This work has been supported by the Klaus Tschira Foundation, Heidelberg, Germany and the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

[15] WikiCoref only contains 30 documents, which is not enough for training neural coreference resolvers.

References

Alan Agresti. 2007. *An Introduction to Categorical Data Analysis.* John Wiley & Sons.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Iyad Batal and Milos Hauskrecht. 2010. Constructing classification features using minimal predictive patterns. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 869–878.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 294–303.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 49–55.

Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. 2007. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007)*, pages 716–725.

Hong Cheng, Xifeng Yan, Jiawei Han, and Philip S Yu. 2008. Direct discriminative pattern mining for effective classification. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE 2008)*, pages 169–178.

Kevin Clark and Christopher D. Manning. 2016a. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016.

Kevin Clark and Christopher D. Manning. 2016b. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 2256–2262.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 660–669.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings ACL-HLT 2008 Conference Short Papers,* Columbus, Ohio, 15–20 June 2008, pages 41–44.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Eraldo R Fernandes and Ruy L Milidiú. 2012. Entropy-guided feature generation for structured learning of Portuguese dependency parsing. In *Proceedings of the International Conference on Computational Processing of the Portuguese Language*, pages 146–156. Springer.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 41–48.

Abbas Ghaddar and Philippe Langlais. 2016a. Coreference in Wikipedia: Main concept resolution. In *Proceedings of the 20th Conference on Computational Natural Language Learning,* Berlin, Germany, 7–11 August 2016, pages 229–238.

Abbas Ghaddar and Philippe Langlais. 2016b. Wiki-Coref: An English coreference-annotated corpus of Wikipedia articles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation,* Portorož, Slovenia, 23–28 May 2016.

Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1-5):53–87.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence,* Edinburgh, Scotland, 30 July – 5 August 2005, pages 1085–1090.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark.

M. Lichman. 2013. UCI machine learning repository.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

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.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 632–642.

Nafise Sadat Moosavi and Michael Strube. 2017a. Lexical features in coreference resolution: To be used with caution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Vancouver, B.C., Canada, 30 July –4 August 2017.

Nafise Sadat Moosavi and Michael Strube. 2017b. Use generalized representations, but do not forget surface features. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 1–7, Valencia, Spain.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Eric W. Noreen. 1989. *Computer Intensive Methods for Hypothesis Testing: An Introduction*. Wiley, New York, N.Y.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,* New York, N.Y., 4–9 June 2006, pages 192–199.

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),* Baltimore, Md., 22–27 June 2014, pages 30–35.

Marta Recasens and Eduard Hovy. 2009. A deeper look into features for coreference resolution. In *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution*, pages 29–42.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France.

Marc Segond and Christian Borgelt. 2011. Item set mining based on cover similarity. *Advances in Knowledge Discovery and Data Mining*, pages 493–505.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Olga Uryupina. 2007. *Knowledge acquisition for coreference resolution*. Ph.D. thesis, Saarland University.

Olga Uryupina and Alessandro Moschitti. 2015. A state-of-the-art mention-pair model for coreference resolution. In *Proceedings of STARSEM 2015: The Fourth Joint Conference on Lexical and Computational Semantics,* Denver, Col., 4–5 June 2015, pages 289–298.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Geoffrey I. Webb. 2006. Discovering significant patterns. *Machine Learning*, 68(1):1–39.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1416–1426.

Xiaofeng Yang, Jian Su, Guodung Zhou, and Chew Lim Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 128–135.

Dian Yu and Heng Ji. 2016. Unsupervised person slot filling based on graph mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 44–53.