BESTCUT: A Graph Algorithm for Coreference Resolution

Cristina Nicolae and Gabriel Nicolae
Human Language Technology Research Institute
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688
{cristina, gabriel}@hlt.utdallas.edu

Abstract

In this paper we describe a coreference resolution method that employs a classification and a clusterization phase. In a novel way, the clusterization is produced as a graph cutting algorithm, in which nodes of the graph correspond to the mentions of the text, whereas the edges of the graph constitute the confidences derived from the coreference classification. In experiments, the graph cutting algorithm for coreference resolution, called BESTCUT, achieves state-of-the-art performance.

1 Introduction

Recent coreference resolution algorithms tackle the problem of identifying coreferent mentions of the same entity in text as a two step procedure: (1) a classification phase that decides whether pairs of noun phrases corefer or not; and (2) a clusterization phase that groups together all mentions that refer to the same entity. An **entity** is an object or a set of objects in the real world, while a **mention** is a textual reference to an entity[1]. Most of the previous coreference resolution methods have similar classification phases, implemented either as decision trees (Soon et al., 2001) or as maximum entropy classifiers (Luo et al., 2004). Moreover, these methods employ similar feature sets. The clusterization phase is different across current approaches. For example, there are several linking decisions for clusterization. (Soon et al., 2001) advocate the link-first decision, which links a mention to its closest candidate referent, while (Ng and Cardie, 2002) consider instead the link-best decision, which links a mention to its most confident candidate referent. Both these clustering decisions are locally optimized. In contrast, globally optimized clustering decisions were reported in (Luo et al., 2004) and (DaumeIII and Marcu, 2005a), where all clustering possibilities are considered by searching on a Bell tree representation or by using the *Learning as Search Optimization (LaSO)* framework (DaumeIII and Marcu, 2005b) respectively, but the first search is partial and driven by heuristics and the second one only looks back in text. We argue that a more adequate clusterization phase for coreference resolution can be obtained by using a graph representation.

In this paper we describe a novel representation of the coreference space as an undirected edge-weighted graph in which the nodes represent all the mentions from a text, whereas the edges between nodes constitute the confidence values derived from the coreference classification phase. In order to detect the entities referred in the text, we need to partition the graph such that all nodes in each subgraph refer to the same entity. We have devised a graph partitioning method for coreference resolution, called BESTCUT, which is inspired from the well-known graph-partitioning algorithm Min-Cut (Stoer and Wagner, 1994). BESTCUT has a different way of computing the cut weight than Min-Cut and a different way of stopping the cut[2]. Moreover, we have slightly modified the Min-Cut procedures. BESTCUT replaces the bottom-up search in a tree representation (as it was performed in (Luo et al., 2004)) with the top-down problem of obtaining the best partitioning of a graph. We start by assuming that all mentions refer to a single entity; the graph cut splits the mentions into subgraphs and the split-

[1] This definition was introduced in (NIST, 2003).

[2] Whenever a graph is split in two subgraphs, as defined in (Cormen et al., 2001), a cut of the graph is produced.

Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pages 275–283,
Sydney, July 2006.

ting continues until each subgraph corresponds to one of the entities. The cut stopping decision has been implemented as an SVM-based classification (Cortes and Vapnik, 1995).

The classification and clusterization phases assume that all mentions are detected. In order to evaluate our coreference resolution method, we have (1) implemented a mention detection procedure that has the novelty of employing information derived from the word senses of common nouns as well as selected lexico-syntactic information; and (2) used a maximum entropy model for coreference classification. The experiments conducted on MUC and ACE data indicate state-of-the-art results when compared with the methods reported in (Ng and Cardie, 2002) and (Luo et al., 2004).

The remainder of the paper is organized as follows. In Section 2 we describe the coreference resolution method that uses the BESTCUT clusterization; Section 3 describes the approach we have implemented for detecting mentions in texts; Section 4 reports on the experimental results; Section 5 discusses related work; finally, Section 6 summarizes the conclusions.

2 BESTCUT Coreference Resolution

For each entity type (PERSON, ORGANIZATION, LOCATION, FACILITY or GPE[3]) we create a graph in which the nodes represent all the mentions of that type in the text, the edges correspond to all pairwise coreference relations, and the edge weights are the confidences of the coreference relations. We will divide this graph repeatedly by cutting the links between subgraphs until a stop model previously learned tells us that we should stop the cutting. The end result will be a partition that approximates the correct division of the text into entities.

We consider this graph approach to clustering a more accurate representation of the relations between mentions than a tree-based approach that treats only anaphora resolution, trying to connect mentions with candidate referents that appear in text before them. We believe that a correct resolution has to tackle cataphora resolution as well, by taking into account referents that appear in the text after the anaphors. Furthermore, we believe that a graph representation of mentions in a text is more adequate than a tree representation because the coreference relation is symmetrical in addition to being transitive. A greedy bottom-up approach does not make full use of this property. A graph-based clusterization starts with a complete overall view of all the connections between mentions, therefore local errors are much less probable to influence the correctness of the outcome. If two mentions are strongly connected, and one of them is strongly connected with the third, all three of them will most probably be clustered together even if the third edge is not strong enough, and that works for any order in which the mentions might appear in the text.

2.1 Learning Algorithm

The coreference confidence values that become the weights in the starting graphs are provided by a maximum entropy model, trained on the training datasets of the corpora used in our experiments. For maximum entropy classification we used a *maxent*[4] tool. Based on the data seen, a maximum entropy model (Berger et al., 1996) offers an expression (1) for the probability that there exists coreference C between a mention m_i and a mention m_j.

$$P(C|m_i, m_j) = \frac{e^{(\sum_k \lambda_k g_k(m_i, m_j, C))}}{Z(m_i, m_j)} \quad (1)$$

where $g_k(m_i, m_j, C)$ is a feature and λ_k is its weight; $Z(m_i, m_j)$ is a normalizing factor.

We created the training examples in the same way as (Luo et al., 2004), by pairing all mentions of the same type, obtaining their feature vectors and taking the outcome (coreferent/non-coreferent) from the key files.

2.2 Feature Representation

We duplicated the statistical model used by (Luo et al., 2004), with three differences. First, no feature combination was used, to prevent long running times on the large amount of ACE data. Second, through an analysis of the validation data, we implemented seven new features, presented in Table 1. Third, as opposed to (Luo et al., 2004), who represented all numerical features quantized, we translated each numerical feature into a set of binary features that express whether the value is in certain intervals. This transformation was necessary because our maximum entropy tool performs better on binary features. (Luo et al., 2004)'s features were not reproduced here from lack of space; please refer to the relevant paper for details.

[3] Entity types as defined by (NIST, 2003).

[4] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

Category	*Feature name*	*Feature description*
lexical	*head-match*	**true** if the two heads are identical
	type-pair	for each mention: name → its type, noun → _NOUN_, pronoun → its spelling
	name-alias	**true** if a mention is an alias of the other one
syntactic	*same-governing-category*	**true** if both mentions are covered by the same type of node, e.g. NP, VP, PP
	path	the parse tree path from m_2 to m_1
	coll-comm	**true** if either mention collocates with a communication verb
grammatical	*gn-agree*	**true** if the two mentions agree in gender and number

Table 1: The added features for the coreference model.

2.3 Clusterization Method: BESTCUT

We start with five initial graphs, one for each entity type, each containing all the mentions of that type and their weighted connections. This initial division is correct because no mentions of different entity types will corefer. Furthermore, by doing this division we avoid unnecessary confusion in the program's decisions and we decrease its running time. Each of these initial graphs will be cut repeatedly until the resulting partition is satisfactory. In each cut, we eliminate from the graph the edges between subgraphs that have a very weak connection, and whose mentions are most likely not part of the same entity.

Formally, the graph model can be defined as follows. Let $M = \{m_i : 1..n\}$ be n mentions in the document and $E = \{e_j : 1..m\}$ be m entities. Let $g : M \rightarrow E$ be the map from a mention $m_i \in M$ to an entity $e_j \in E$. Let $c : M\text{x}M \rightarrow [0, 1]$ be the confidence the learning algorithm attaches to the coreference between two mentions $m_i, m_j \in M$. Let $T = \{t_k : 1..p\}$ be the set of entity types or classes. Then we attach to each entity class t_k an undirected, edge-weighted graph $G_k(V_k, E_k)$, where $V_k = \{m_i | g(m_i).type = t_k\}$ and $E_k = \{(m_i, m_j, c(m_i, m_j)) | m_i, m_j \in V_k\}$.

The partitioning of the graph is based at each step on the cut weight. As a starting point, we used the Min-Cut algorithm, presented and proved correct in (Stoer and Wagner, 1994). In this simple and efficient method, the weight of the cut of a graph into two subgraphs is the sum of the weights of the edges crossing the cut. The partition that minimizes the cut weight is the one chosen. The main procedure of the algorithm computes cuts-of-the-phase repeatedly and selects the one with the minimum cut value (cut weight). We adapted this algorithm to our coreference situation.

To decide the minimum cut (from here on called the BESTCUT), we use as cut weight the number of mentions that are correctly placed in their set. The method for calculating the correctness score is presented in Figure 1. The BESTCUT at one stage is the cut-of-the-phase with the highest correctness score.

```
cut-weight(Graph G, Cut C = (S,T))
1 corrects-avg ← corrects-max ← 0
2 foreach m ∈ G.V
3   if m ∈ S.V then setm ← S
4   else setm ← T
7   if avg_{n∈setm.V,n≠m} weight(m,n) >
       avg_{n∈G.V\setm.V} weight(m,n)
6   then corrects-avg++
7   if max_{n∈setm.V,n≠m} weight(m,n) >
       max_{n∈G.V\setm.V} weight(m,n)
8   then corrects-max++
9 return (corrects-avg +
          corrects-max) / 2
```

Figure 1: Computing the cut-weight.

An additional learning model was trained to decide if cutting a set of mentions is better or worse than keeping the mentions together. The model was optimized to maximize the ECM-F score[5]. We will denote by S the larger part of the cut and T the smaller one. $C.E$ is the set of edges crossing the cut, and G is the current graph before the cut. $S.V$ and $T.V$ are the set of vertexes in S and in T, respectively. $S.E$ is the set of edges from S, while $T.E$ is the set of edges from T. The features for stopping the cut are presented in Table 2. The model was trained using 10-fold cross-validation on the training set. In order to learn when to stop the cut, we generated a list of positive and negative examples from the training files. Each training example is associated with a certain cut (S, T). Since we want to learn a stop function, the positive examples must be examples that describe when the cut must not be done, and the negative examples are examples that present situations when the cut must be performed. Let us consider that the list of entities from a text is $E = \{e_j : 1..m\}$ with $e_j = \{m_{i_1}, m_{i_2}, ...m_{i_k}\}$ the list of mentions that refer to e_j. We generated a negative example for each pair $(S = \{e_i\}, T = \{e_j\})$ with $i \neq j$ – each entity must be separated from any other en-

[5] As introduced by (Luo et al., 2004).

Feature name	*Feature description*
st-ratio	$\|S.V\|/\|T.V\|$ – the ratio between the cut parts
ce-ratio	$\|C.E\|/\|G.E\|$ – the proportion of the cut from the entire graph
c-min	$min(C.E)$ – the smallest edge crossing the cut
c-max	$max(C.E)$ – the largest edge crossing the cut
c-avg	$avg(C.E)$ – the average of the edges crossing the cut
c-hmean	$hmean(C.E)$ – the harmonic mean of the edges crossing the cut
c-hmeax	$hmeax(C.E)$ – a variant of the harmonic mean. $hmeax(C.E) = 1 - hmean(C.E')$ where each edge from E' has the weight equal to 1 minus the corresponding edge from E
lt-c-avg-ratio	how many edges from the cut are less than the average of the cut (as a ratio)
lt-c-hmean-ratio	how many edges from the cut are less than the harmonic mean of the cut (as a ratio)
st-avg	$avg(S.E + T.E)$ – the average of the edges from the graph when the edges from the cut are not considered
g-avg	$avg(G.E)$ – the average of the edges from the graph
st-wrong-avg-ratio	how many vertexes are in the wrong part of the cut using the average measure for the 'wrong' (as a ratio)
st-wrong-max-ratio	how many vertexes are in the wrong part of the cut using the max measure for the 'wrong' (as a ratio)
lt-c-avg-ratio $<$ *st-lt-c-avg-ratio*	1 if $r_1 < r_2$, 0 otherwise; r_1 is the ratio of the edges from $C.E$ that are smaller than the average of the cut; r_2 is the ratio of the edges from $S.E + T.E$ that are smaller than the average of the cut
g-avg $>$ *st-avg*	1 if the $avg(G.E) > avg(S.E + T.E)$, and 0 otherwise

Table 2: The features for stopping the cut.

tity. We also generated negative examples for all pairs $(S = \{e_i\}, T = E \setminus S)$ – each entity must be separated from all the other entities considered together. To generate positive examples, we simulated the cut on a graph corresponding to a single entity e_j. Every partial cut of the mentions of e_j was considered as a positive example for our stop model.

We chose not to include pronouns in the BESTCUT initial graphs, because, since most features are oriented towards Named Entities and common nouns, the learning algorithm (*maxent*) links pronouns with very high probability to many possible antecedents, of which not all are in the same chain. Thus, in the clusterization phase the pronouns would act as a bridge between different entities that should not be linked. To prevent this, we solved the pronouns separately (at the end of the BESTCUT algorithm) by linking them to their antecedent with the best coreference confidence.

```
BESTCUT(Graph G_i)
 1 entities.clear()
 2 queue.push_back(G_i)
 3 while not queue.empty()
 4   G ← queue.pop_front()
 5   (S,T) ← ProposeCut(G)
 6   if StopTheCut(G,S,T)
 7   then
 8     entities.add(NewEntity(G))
 9   else
10     queue.push_back(S)
11     queue.push_back(T)
12 return entities
```

Figure 2: The general algorithm for BESTCUT.

Figure 2 details the main procedure of the BESTCUT algorithm. The algorithm receives as input a weighted graph having a vertex for each mention considered and outputs the list of entities created. In each stage, a cut is proposed for all subgraphs in the queue. In case StopTheCut decides that the cut must be performed on the subgraph, the two sides of the cut are added to the queue (lines 10-11); if the graph is well connected and breaking the graph in two parts would be a bad thing, the current graph will be used to create a single entity (line 8). The algorithm ends when the queue becomes empty. ProposeCut (Figure 3) returns a cut of the graph obtained with an algorithm similar to the Min-Cut algorithm's procedure called MinimumCut. The differences between our algorithm and the Min-Cut procedure are that **the most tightly connected vertex** in each step of the ProposeCutPhase procedure, z, is found using expression 2:

```
ProposeCut(Graph G)
1 while |G.V| > 1
2   (S,T) ← ProposeCutPhase(G)
3   if the cut-of-the-phase (S,T)
       is-lighter than the current
       best cut (S_b,T_b)
4   then store the cut-of-the-phase
         as (S_b,T_b)
5 return (S_b,T_b)
```

Figure 3: The algorithm for ProposeCut.

$$z = argmax_{y \notin A} w_a(A, y) \quad (2)$$

where $w_a(A, y) = \frac{1}{|A|} \sum_{x \in A} w(x, y)$, and the **is-lighter** test function uses the correctness score presented before: the partial cut with the larger correctness score is better. The ProposeCutPhase function is presented in Figure 4.

```
ProposeCutPhase(Graph G)
1 A ← {G.V.first}
2 while |A| < |G.V|
3   last ← the most tightly
         connected vertex
4   add last to A
5 store the cut-of-the-phase and
  shrink G by merging the two
  vertexes added last
6 return (G.V \ {last}, last)
```

Figure 4: The algorithm for ProposeCutPhase.

2.4 An Example

Let us consider an example of how the BESTCUT algorithm works on two simple sentences (Figure 5). The entities present in this example are: {*Mary*$_1$, *the girl*$_5$} and {*a brother*$_2$, *John*$_3$, *The boy*$_4$}. Since they are all PERSONs, the algorithm

> ***Mary***$_1$ *has* ***a brother***$_2$, ***John***$_3$. ***The boy***$_4$ *is older than* ***the girl***$_5$.

Figure 5: An example.

will be applied on a single graph, corresponding to the class PERSON and composed of all these mentions.

The initial graph is illustrated in Figure 6, with the coreference relation marked through a different coloring of the nodes. Each node number corresponds to the mention with the same index in Figure 5.

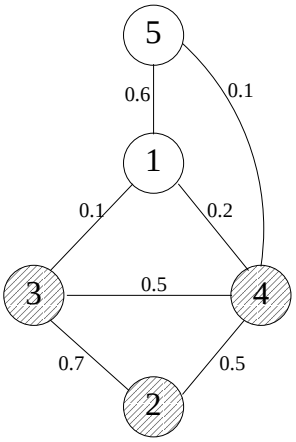

Figure 6: The initial graph

The strongest confidence score is between *a brother*$_2$ and *John*$_3$, because they are connected through an apposition relation. The graph was simplified by eliminating the edges that have an insignificant weight, e.g. the edges between *John*$_3$ and *the girl*$_5$ or between *Mary*$_1$ and *a brother*$_2$.

Function BESTCUT starts with the whole graph. The first cut of the phase, obtained by function ProposeCutPhase, is the one in Figure 7.a. This

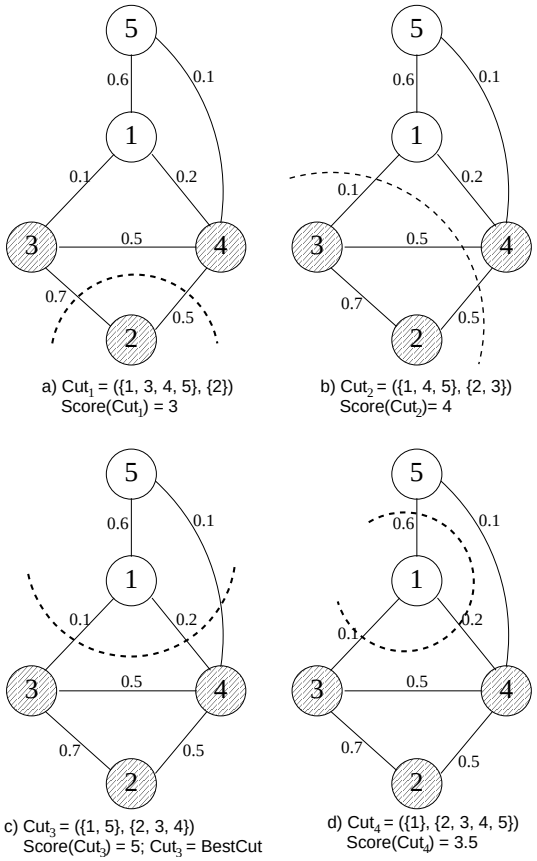

Figure 7: Cuts-of-the-phase

cut separates node 2 from the rest of the graph. In calculating the score of the cut (using the algorithm from Figure 1), we obtain an average number of three correctly placed mentions. This can be verified intuitively on the drawing: mentions 1, 2 and 5 are correctly placed, while 3 and 4 are not. The score of this cut is therefore 3. The second, the third and the fourth cuts of the phase, in Figures 7.b, 7.c and 7.d, have the scores 4, 5 and 3.5 respectively. An interesting thing to note at the fourth cut is that the score is no longer an integer. This happens because it is calculated as an average between *corrects-avg = 4* and *corrects-max = 3*. The methods disagree about the placement of mention 1. The average of the outgoing weights of mention 1 is 0.225, less than 0.5 (the default weight assigned to a single mention) therefore the first method declares it is correctly placed. The second considers only the maximum; 0.6 is greater than 0.5, so the mention appears to be more strongly connected with the outside than the inside. As we can see, the contradiction is because of the uneven distribution of the weights of the outgoing edges.

The first proposed cut is the cut with the great-

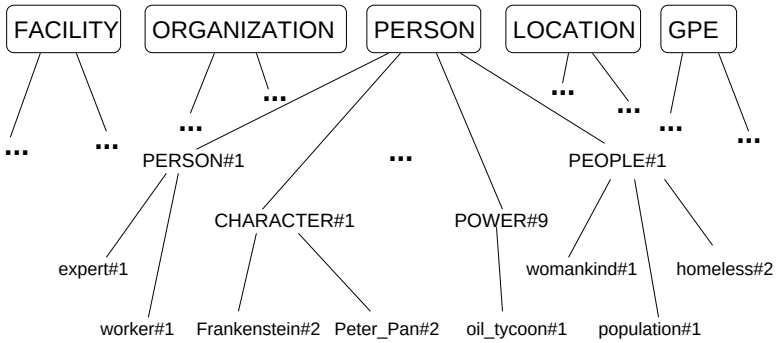

Figure 8: Part of the hierarchy containing 42 WordNet equivalent concepts for the five entity types, with all their synonyms and hyponyms. The hierarchy has 31,512 word-sense pairs in total

est score, which is Cut_3 (Figure 7.c). Because this is also the correct cut, all cuts proposed after this one will be ignored– the machine learning algorithm that was trained when to stop a cut will always declare against further cuts. In the end, the cut returned by function BESTCUT is the correct one: it divides mentions *Mary*$_1$ and *the girl*$_5$ from mentions *a brother*$_2$, *John*$_3$ and *The boy*$_4$.

3 Mention Detection

Because our BESTCUT algorithm relies heavily on knowing entity types, we developed a method for recognizing entity types for nominal mentions. Our statistical approach uses maximum entropy classification with a few simple lexical and syntactic features, making extensive use of WordNet (Fellbaum, 1998) hierarchy information. We used the ACE corpus, which is annotated with mention and entity information, as data in a supervised machine learning method to detect nominal mentions and their entity types. We assigned six entity types: PERSON, ORGANIZATION, LOCATION, FACILITY, GPE and UNK (for those who are in neither of the former categories) and two genericity outcomes: GENERIC and SPECIFIC. We only considered the intended value of the mentions from the corpus. This was motivated by the fact that we need to classify mentions according to the context in which they appear, and not in a general way. Only contextual information is useful further in coreference resolution. We have experimentally discovered that the use of word sense disambiguation improves the performance tremendously (a boost in score of 10%), therefore all the features use the word senses from a previously-applied word sense disambiguation program, taken from (Mihalcea and Csomai, 2005).

For creating training instances, we associated an outcome to each markable (NP) detected in the training files: the markables that were present in the key files took their outcome from the key file annotation, while all the other markables were associated with outcome UNK. We then created a training example for each of the markables, with the feature vector described below and as target function the outcome. The aforementioned outcome can be of three different types. The first type of outcome that we tried was the entity type (one member of the set PERSON, ORGANIZATION, LOCATION, FACILITY, GPE and UNK); the second type was the genericity information (GENERIC or SPECIFIC), whereas the third type was a combination between the two (pairwise combinations of the entity types set and the genericity set, e.g. PERSON_SPECIFIC).

The feature set consists of WordNet features, lexical features, syntactic features and intelligent context features, briefly described in Table 3. With the WordNet features we introduce the *WordNet equivalent concept*. A WordNet equivalent concept for an entity type is a word-sense pair from WordNet whose gloss is compatible with the definition of that entity type. Figure 8 enumerates a few WordNet equivalent concepts for entity class PERSON (e.g. CHARACTER#1), with their hierarchy of hyponyms (e.g. Frankenstein#2). The lexical feature is useful because some words are almost always of a certain type (e.g. *"company"*). The intelligent context set of features are an improvement on basic context features that use the stems of the words that are within a window of a certain size around the word. In addition to this set of features, we created more features by combining them into pairs. Each pair contains two features from two different classes. For instance, we will have features like: *is-a-*

Category	*Feature name*	*Feature description*
WordNet	is-a-TYPE	**true** if the mention is of entity type TYPE; five features
	WN-eq-concept-hyp	**true** if the mention is in hyponym set of WN-*eq-concept*; 42 features
	WN-eq-concept-syn	**true** if the mention is in synonym set of WN-*eq-concept*; 42 features
lexical	*stem-sense*	pair between the stem of the word and the WN sense of the word by the WSD
syntactic	*pos*	part of speech of the word by the POS tagger
	is-modifier	**true** if the mention is a modifier in another noun phrase
	*modifier-to-*TYPE	**true** if the mention is a modifier to a TYPE mention
	in-apposition-with	TYPE of the mention our mention is in apposition with
intelligent context	*all-mods*	the nominal, adjectival and pronominal modifiers in the mention's parse tree
	preps	the prepositions right before and after the mention's parse tree

Table 3: The features for the mention detection system.

PERSON∼*in-apposition-with(*PERSON*)*.

All these features apply to the "true head" of a noun phrase, i.e. if the noun phrase is a partitive construction (*"five students"*, *"a lot of companies"*, *"a part of the country"*), we extract the "true head", the whole entity that the part was taken out of (*"students"*, *"companies"*, *"country"*), and apply the features to that "true head" instead of the partitive head.

For combining the mention detection module with the BESTCUT coreference resolver, we also generated classifications for Named Entities and pronouns by using the same set of features minus the WordNet ones (which only apply to nominal mentions). For the Named Entity classifier, we added the feature *Named-Entity-type* as obtained by the Named Entity Recognizer. We generated a list of all the markable mentions and their entity types and presented it as input to the BESTCUT resolver instead of the list of perfect mentions. Note that this mention detection does not contain complete anaphoricity information. Only the mentions that are a part of the five considered classes are treated as anaphoric and clustered, while the UNK mentions are ignored, even if an outside anaphoricity classifier might categorize some of them as anaphoric.

4 Experimental Results

The clusterization algorithms that we implemented to evaluate in comparison with our method are (Luo et al., 2004)'s Belltree and Link-Best (best-first clusterization) from (Ng and Cardie, 2002). The features used were described in section 2.2. We experimented on the ACE Phase 2 (NIST, 2003) and MUC6 (MUC-6, 1995) corpora. Since we aimed to measure the performance of coreference, the metrics used for evaluation are the ECM-F (Luo et al., 2004) and the MUC P, R and F scores (Vilain et al., 1995).

In our first experiment, we tested the three coreference clusterization algorithms on the development-test set of the ACE Phase 2 corpus, first on true mentions (i.e. the mentions annotated in the key files), then on detected mentions (i.e. the mentions output by our mention detection system presented in section 3) and finally without any prior knowledge of the mention types. The results obtained are tabulated in Table 4. As can be observed, when it has prior knowledge of the mention types BESTCUT performs significantly better than the other two systems in the ECM-F score and slightly better in the MUC metrics. The more knowledge it has about the mentions, the better it performs. This is consistent with the fact that the first stage of the algorithm divides the graph into subgraphs corresponding to the five entity types. If BESTCUT has no information about the mentions, its performance ranks significantly under the Link-Best and Belltree algorithms in ECM-F and MUC R. Surprisingly enough, the Belltree algorithm, a globally optimized algorithm, performs similarly to Link-Best in most of the scores.

Despite not being as dramatically affected as BESTCUT, the other two algorithms also decrease in performance with the decrease of the mention information available, which empirically proves that mention detection is a very important module for coreference resolution. Even with an F-score of 77.2% for detecting entity types, our mention detection system boosts the scores of all three algorithms when compared to the case where no information is available.

It is apparent that the MUC score does not vary significantly between systems. This only shows that none of them is particularly poor, but it is not a relevant way of comparing methods– the MUC metric has been found too indulgent by researchers ((Luo et al., 2004), (Baldwin et al., 1998)). The MUC scorer counts the common links between the

Clusterization algorithm	*Mentions*	ECM-F%	MUC *score*		
			MUC P%	MUC R%	MUC F%
BESTCUT	key	**82.7**	**91.1**	88.2	**89.63**
	detected	**73.0**	**88.3**	75.1	**81.17**
	undetected	41.2	**52.0**	82.4	**63.76**
Belltree (Luo et al., 2004)	key	77.9	88.5	89.3	88.90
	detected	70.8	86.0	76.6	81.03
	undetected	**52.6**	40.3	87.1	55.10
Link-Best (Ng and Cardie, 2002)	key	77.9	88.0	**90.0**	88.99
	detected	70.7	85.1	**77.3**	81.01
	undetected	51.6	39.6	**88.5**	54.72

Table 4: Comparison of results between three clusterization algorithms on ACE Phase 2. The learning algorithms are *maxent* for coreference and SVM for stopping the cut in BESTCUT. In turn, we obtain the mentions from the key files, detect them with our mention detection algorithm or do not use any information about them.

annotation keys and the system output, while the ECM-F metric aligns the detected entities with the key entities so that the number of common mentions is maximized. The ECM-F scorer overcomes two shortcomings of the MUC scorer: not considering single mentions and treating every error as equally important (Baldwin et al., 1998), which makes the ECM-F a more adequate measure of coreference.

Our second experiment evaluates the impact that the different categories of our added features have on the performance of the BESTCUT system. The experiment was performed with a maxent classifier on the MUC6 corpus, which was priorly converted into ACE format, and employed mention information from the key annotations.

Model	ECM-F%	MUC *score*		
		P%	R%	F%
baseline	78.3	89.5	91.5	90.49
+grammatical	78.4	89.2	92.5	90.82
+lexical	83.1	92.4	91.6	92.00
+syntactic	**85.1**	**92.7**	**92.4**	**92.55**

Table 5: Impact of feature categories on BESTCUT on MUC6. Baseline system has the (Luo et al., 2004) features. The system was tested on key mentions.

From Table 5 we can observe that the lexical features (*head-match*, *type-pair*, *name-alias*) have the most influence on the ECM-F and MUC scores, succeeded by the syntactic features (*same-governing-category*, *path*, *coll-comm*). Despite what intuition suggests, the improvement the grammatical feature *gn-agree* brings to the system is very small.

5 Related Work

It is of interest to discuss why our implementation of the Belltree system (Luo et al., 2004) is comparable in performance to Link-Best (Ng and Cardie, 2002). (Luo et al., 2004) do the clusterization through a beam-search in the Bell tree using either a mention-pair or an entity-mention model, the first one performing better in their experiments. Despite the fact that the Bell tree is a complete representation of the search space, the search in it is optimized for size and time, while potentially losing optimal solutions– similarly to a Greedy search. Moreover, the fact that the two implementations are comparable is not inconceivable once we consider that (Luo et al., 2004) never compared their system to another coreference resolver and reported their competitive results on true mentions only.

(Ng, 2005) treats coreference resolution as a problem of ranking candidate partitions generated by a set of coreference systems. The overall performance of the system is limited by the performance of its best component. The main difference between this approach and ours is that (Ng, 2005)'s approach takes coreference resolution one step further, by comparing the results of multiple systems, while our system is a single resolver; furthermore, he emphasizes the global optimization of ranking clusters obtained locally, whereas our focus is on globally optimizing the clusterization method inside the resolver.

(DaumeIII and Marcu, 2005a) use the *Learning as Search Optimization* framework to take into account the non-locality behavior of the coreference features. In addition, the researchers treat mention detection and coreference resolution as a joint problem, rather than a pipeline approach like we

do. By doing so, it may be easier to detect the entity type of a mention once we have additional clues (expressed in terms of coreference features) about its possible antecedents. For example, labeling *Washington* as a PERSON is more probable after encountering *George Washington* previously in the text. However, the coreference problem does not immediately benefit from the joining.

6 Conclusions

We have proposed a novel coreference clusterization method that takes advantage of the efficiency and simplicity of graph algorithms. The approach is top-down and globally optimized, and takes into account cataphora resolution in addition to anaphora resolution. Our system compares favorably to two other implemented coreference systems and achieves state-of-the-art performance on the ACE Phase 2 corpus on true and detected mentions. We have also briefly described our mention detection system whose output we used in conjunction with the BESTCUT coreference system to achieve better results than when no mention information was available.

Acknowledgments

We would like to thank the three anonymous reviewers for their very helpful suggestions and comments on the early draft of our paper.

References

B. Baldwin, T. Morton, A. Bagga, J. Baldridge, R. Chandraseker, A. Dimitriadis, K. Snyder, and M. Wolska. 1998. Description of the university of pennsylvania camp system as used for coreference. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*.

A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 1(22):39–71.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms, Second Edition*. MIT.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

H. DaumeIII and D. Marcu. 2005a. A large-scale exploration of effective global features for a joint entity detection and tracking model. pages 97–104, Vancouver.

H. DaumeIII and D. Marcu. 2005b. Learning as search optimization: Approximate large margin methods for structured prediction. In *The International Conference on Machine Learning (ICML)*.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

R. Mihalcea and A. Csomai. 2005. Senselearner: Word sense disambiguation for all words in unrestricted text. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, Ann Arbor, MI.

MUC-6. 1995. Coreference task definition.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 4Oth Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania.

V. Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

V. Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, pages 157–164, Ann Arbor, MI.

NIST. 2003. Entity detection and tracking - phase 1; edt and metonymy annotation guidelines. version 2.5.1 20030502.

M. Pasca and S. Harabagiu. 2001. High performance question/answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 366–374, New Orleans, LA.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 4(27):521–544.

M. Stoer and F. Wagner. 1994. A simple min cut algorithm. In Jan van Leeuwen, editor, *Proceedings of the 1994 European Symposium on Algorithms*, pages 141–147, New York. Springer-Verlag.

M. Vilain, J.Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 45–52.