A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree

Xiaoqiang Luo and Abe Ittycheriah
Hongyan Jing and Nanda Kambhatla and Salim Roukos
1101 Kitchawan Road
Yorktown Heights, NY 10598, U.S.A.
{xiaoluo,abei,hjing,nanda,roukos}@us.ibm.com

Abstract

This paper proposes a new approach for coreference resolution which uses the *Bell* tree to represent the search space and casts the coreference resolution problem as finding the best path from the root of the Bell tree to the leaf nodes. A Maximum Entropy model is used to rank these paths. The coreference performance on the 2002 and 2003 Automatic Content Extraction (ACE) data will be reported. We also train a coreference system using the MUC6 data and competitive results are obtained.

1 Introduction

In this paper, we will adopt the terminologies used in the Automatic Content Extraction (ACE) task (NIST, 2003). Coreference resolution in this context is defined as partitioning mentions into entities. A *mention* is an instance of reference to an object, and the collection of mentions referring to the same object in a document form an *entity*. For example, in the following sentence, mentions are underlined:

> "The American Medical Association voted yesterday to install the heir apparent as its president-elect, rejecting a strong, upstart challenge by a District doctor who argued that the nation's largest physicians' group needs stronger ethics and new leadership."

"American Medical Association", "its" and "group" belong to the same entity as they refer to the same object.

Early work of anaphora resolution focuses on finding antecedents of pronouns (Hobbs, 1976; Ge et al., 1998; Mitkov, 1998), while recent advances (Soon et al., 2001; Yang et al., 2003; Ng and Cardie, 2002; Ittycheriah et al., 2003) employ statistical machine learning methods and try to resolve reference among all kinds of noun phrases (NP), be it a name, nominal, or pronominal phrase – which is the scope of this paper as well. One common strategy shared by (Soon et al., 2001; Ng and Cardie, 2002; Ittycheriah et al., 2003) is that a statistical model is trained to measure how likely a pair of mentions corefer; then a greedy procedure is followed to group mentions into entities. While this approach has yielded encouraging results, the way mentions are linked is arguably suboptimal in that an instant decision is made when considering whether two mentions are linked or not.

In this paper, we propose to use the *Bell tree* to represent the process of forming entities from mentions. The Bell tree represents the search space of the coreference resolution problem – each leaf node corresponds to a possible coreference outcome. We choose to model the process from mentions to entities represented in the Bell tree, and the problem of coreference resolution is cast as finding the "best" path from the root node to leaves. A binary maximum entropy model is trained to compute the linking probability between a partial entity and a mention.

The rest of the paper is organized as follows. In Section 2, we present how the Bell tree can be used to represent the process of creating entities from mentions and the search space. We use a maximum entropy model to rank paths in the Bell tree, which is discussed in Section 3. After presenting the search strategy in Section 4, we show the experimental results on the ACE 2002 and 2003 data, and the Message Understanding Conference (MUC) (MUC, 1995) data in Section 5. We compare our approach with some recent work in Section 6.

2 Bell Tree: From Mention to Entity

Let us consider traversing mentions in a document from beginning (left) to end (right). The process of forming entities from mentions can be represented by a tree structure. The root node is the initial state of the process, which consists of a partial entity containing the first mention of a document. The second mention is

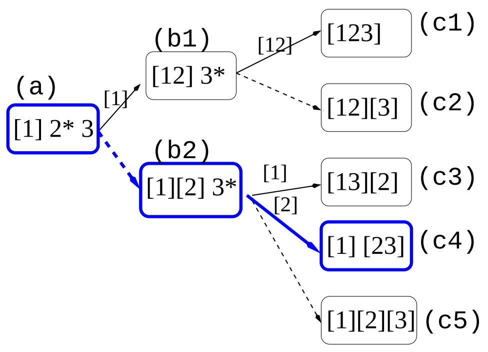

Figure 1: Bell tree representation for three mentions: numbers in [] denote a partial entity. In-focus entities are marked on the solid arrows, and active mentions are marked by *. Solid arrows signify that a mention is linked with an in-focus partial entity while dashed arrows indicate starting of a new entity.

added in the next step by either *linking* to the existing entity, or *starting* a new entity. A second layer of nodes are created to represent the two possible outcomes. Subsequent mentions are added to the tree in the same manner. The process is *mention-synchronous* in that each layer of tree nodes are created by adding one mention at a time. Since the number of tree leaves is the number of possible coreference outcomes and it equals the Bell Number (Bell, 1934), the tree is called the *Bell* tree. The Bell Number $B(n)$ is the number of ways of partitioning n distinguishable objects (i.e., mentions) into non-empty disjoint subsets (i.e., entities). The Bell Number has a "closed" formula $B(n) = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$ and it increases rapidly as n increases: $B(20) \approx 5.2 \times 10^{13}$! Clearly, an efficient search strategy is necessary, and it will be addressed in Section 4.

Figure 1 illustrates how the Bell tree is created for a document with three mentions. The initial node consists of the first partial entity `[1]` (i.e., node (a) in Figure 1). Next, mention `2` becomes *active* (marked by "*" in node (a)) and can either *link* with the partial entity `[1]` and result in a new node (b1), or *start* a new entity and create another node (b2). The partial entity which the active mention considers linking with is said to be *in-focus*. In-focus entities are highlighted on the solid arrows in Figure 1. Similarly, mention `3` will be active in the next stage and can take five possible actions, which create five possible coreference results shown in node (c1) through (c5).

Under the derivation illustrated in Figure 1, each leaf node in the Bell tree corresponds to a possible coreference outcome, and there is no other way to form entities. The Bell tree clearly represents the search space of the coreference resolution problem. The coreference resolution can therefore be cast equivalently as finding the "best" leaf node. Since the search space is large (even for a document with a moderate number of mentions), it is difficult to estimate a distribution over leaves directly. Instead, we choose to model the *process* from mentions to entities, or in other words, score paths from the root to leaves in the Bell tree.

A nice property of the Bell tree representation is that the number of linking or starting steps is the same for all the hypotheses. This makes it easy to rank them using the "local" linking and starting probabilities as the number of factors is the same. The Bell tree representation is also incremental in that mentions are added sequentially. This makes it easy to design a decoder and search algorithm.

3 Coreference Model

3.1 Linking and Starting Model

We use a binary conditional model to compute the probability that an *active* mention `links` with an *in-focus* partial entity. The conditions include all the partially-formed entities before, the focus entity index, and the active mention.

Formally, let $\{m_i : 1 \le i \le n\}$ be n mentions in a document. Mention index i represents the order it appears in the document. Let e_j be an entity, and $g : i \mapsto j$ be the (many-to-one) map from mention index i to entity index j. For an active mention index $k (1 \le k \le n)$, define

$$I_k = \{t : t = g(i), \text{ for some } 1 \le i \le k-1\},$$

the set of indices of the partially-established entities to the left of m_k (note that $I_1 = \emptyset$), and

$$E_k = \{e_t : t \in I_k\},$$

the set of the partially-established entities. The *link* model is then

$$P(L|E_k, m_k, A_k = t), \qquad (1)$$

the probability linking the active mention m_k with the in-focus entity e_t. The random variable A_k takes value from the set I_k and signifies which entity is in focus; L takes binary value and is 1 if m_k links with e_t.

As an example, for the branch from (b2) to (c4) in Figure 1, the active mention is "3", the set of partial entities to the left of "3" is $E_3 = \{[1], [2]\}$, the active entity is the second partial entity "[2]". Probability $P(L = 1|E_3, “3'', A_3 = 2)$ measures how likely mention "3" links with the entity "[2]."

The model $P(L|E_k, m_k, A_k = t)$ only computes how likely m_k links with e_t; It does not say anything about the possibility that m_k starts a new entity. Fortunately, the *starting* probability can be computed using link probabilities (1), as shown now.

Since starting a new entity means that m_k does not link with any entities in E_k, the probability of starting a new entity, $P(L=0|E_k,m_k)$, can be computed as

$$\begin{aligned} & P(L=0|E_k,m_k) \quad (2) \\ =& \sum_{t \in I_k} P(L=0, A_k=t|E_k,m_k) \\ =& 1 - \sum_{t \in I_k} P(A_k=t|E_k,m_k) \cdot \\ & P(L=1|E_k,m_k,A_k=t). \quad (3) \end{aligned}$$

(3) indicates that the probability of starting an entity can be computed using the linking probabilities $P(L=1|E_k,m_k,A_k=t)$, provided that the marginal $P(A_t=t|E_k,m_k)$ is known. In this paper, $P(A_k=t|E_k,m_k)$ is approximated as:

$$P(A_k=t|E_k,m_k) = \begin{cases} 1 & \text{if } t = \arg\max_{i \in I_k} \\ & \quad P(L=1|E_k,m_k,A_k=i) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

With the approximation (4), the starting probability (3) is

$$\begin{aligned} & P(L=0|E_k,m_k) \\ =& 1 - \max_{t \in I_k} P(L=1|E_k,m_k,A_k=t). \quad (5) \end{aligned}$$

The linking model (1) and approximated starting model (5) can be used to score paths in the Bell tree. For example, the score for the path (a)-(b2)-(c4) in Figure 1 is the product of the start probability from (a) to (b2) and the linking probability from (b2) to (c4).

Since (5) is an approximation, not true probability, a constant α is introduced to balance the linking probability and starting probability and the starting probability becomes:

$$P_\alpha(L=0|E_k,m_k) = \alpha P(L=0|E_k,m_k). \quad (6)$$

If $\alpha < 1$, it penalizes creating new entities; Therefore, α is called *start penalty*. The start penalty α can be used to balance entity miss and false alarm.

3.2 Model Training and Features

The model $P(L|E_k,m_k,A_k=t)$ depends on all partial entities E_k, which can be very expensive. After making some modeling assumptions, we can approximate it as:

$$\begin{aligned} & P(L=1|E_k,m_k,A_k=t) \quad (7) \\ \approx & P(L=1|e_t,m_k) \quad (8) \\ \approx & \max_{m \in e_t} P(L=1|m,m_k). \quad (9) \end{aligned}$$

From (7) to (8), entities other than the one in focus, e_t, are assumed to have no influence on the decision of linking m_k with e_t. (9) further assumes that the entity-mention score can be obtained by the maximum mention pair score. The model (9) is very similar to the model in (Morton, 2000; Soon et al., 2001; Ng and Cardie, 2002) while (8) has more conditions.

We use maximum entropy model (Berger et al., 1996) for both the mention-pair model (9) and the entity-mention model (8):

$$P(L|m_i,m_k) = \frac{e^{\left(\sum_k \lambda_k g_k(m_i,m_k,L)\right)}}{Z(m_i,m_k)}, \quad (10)$$

$$P(L|e_t,m_k) = \frac{e^{\left(\sum_k \lambda_k g_k(e_t,m_k,L)\right)}}{Z(e_t,m_k)}, \quad (11)$$

where $g_k(\cdot,\cdot,L)$ is a feature and λ_k is its weight; $Z(\cdot,\cdot)$ is a normalizing factor to ensure that (10) or (11) is a probability. Effective training algorithm exists (Berger et al., 1996) once the set of features $\{g_k(\cdot,\cdot,L)\}$ is selected.

The basic features used in the models are tabulated in Table 1.

Features in the lexical category are applicable to non-pronominal mentions only. Distance features characterize how far the two mentions are, either by the number of tokens, by the number of sentences, or by the number of mentions in-between. Syntactic features are derived from parse trees output from a maximum entropy parser (Ratnaparkhi, 1997). The "Count" feature calculates how many times a mention string is seen. For pronominal mentions, attributes such as gender, number, possessiveness and reflexiveness are also used. Apart from basic features in Table 1, composite features can be generated by taking conjunction of basic features. For example, a distance feature together with reflexiveness of a pronoun mention can help to capture that the antecedent of a reflexive pronoun is often closer than that of a non-reflexive pronoun.

The same set of basic features in Table 1 is used in the entity-mention model, but feature definitions are slightly different. Lexical features, including the acronym features, and the apposition feature are computed by testing any mention in the entity e_t against the active mention m_k. Editing distance for (e_t,m_k) is defined as the minimum distance over any non-pronoun mentions and the active mention. Distance features are computed by taking minimum between mentions in the entity and the active mention.

In the ACE data, mentions are annotated with three levels: NAME, NOMINAL and PRONOUN. For each ACE entity, a canonical mention is defined as the longest NAME mention if available; or if the entity does not have a NAME mention, the most recent NOMINAL mention; if there is no NAME and NOMINAL mention, the most recent pronoun mention. In the entity-mention model, "ncd","spell" and "count" features are computed over the canonical mention of the in-focus entity and the active mention. Conjunction features are used in the entity-mention model too.

The mention-pair model is appealing for its simplicity: features are easy to compute over a pair of men-

Category	Features	Remark
Lexical	exact_strm	1 if two mentions have the same spelling; 0 otherwise
	left_subsm	1 if one mention is a left substring of the other; 0 otherwise
	right_subsm	1 if one mention is a right substring of the other; 0 otherwise
	acronym	1 if one mention is an acronym of the other; 0 otherwise
	edit_dist	quantized editing distance between two mention strings
	spell	pair of actual mention strings
	ncd	number of different capitalized words in two mentions
Distance	token_dist	how many tokens two mentions are apart (quantized)
	sent_dist	how many sentences two mentions are apart (quantized)
	gap_dist	how many mentions in between the two mentions in question (quantized)
Syntax	POS_pair	POS-pair of two mention heads
	apposition	1 if two mentions are appositive; 0 otherwise
Count	count	pair of (quantized) numbers, each counting how many times a mention string is seen
Pronoun	gender	pair of attributes of {female, male, neutral, unknown }
	number	pair of attributes of {singular, plural, unknown}
	possessive	1 if a pronoun is possessive; 0 otherwise
	reflexive	1 if a pronoun is reflexive; 0 otherwise

Table 1: Basic features used in the maximum entropy model.

tions; its drawback is that information outside the mention pair is ignored. Suppose a document has three mentions “Mr. Clinton”, “Clinton” and “she”, appearing in that order. When considering the mention pair “Clinton” and “she”, the model may tend to link them because of their proximity; But this mistake can be easily avoided if “Mr. Clinton” and “Clinton” have been put into the same entity and the model knows “Mr. Clinton” referring to a male while “she” is female. Since gender and number information is propagated at the entity level, the entity-mention model is able to check the gender consistency when considering the active mention “she”.

3.3 Discussion

There is an in-focus entity in the condition of the linking model (1) while the starting model (2) conditions on all left entities. The disparity is intentional as the *starting* action is influenced by all established entities on the left.

(4) is not the only way $P(A_k = t|E_k, m_k)$ can be approximated. For example, one could use a uniform distribution over I_k. We experimented several schemes of approximation, including a uniform distribution, and (4) worked the best and is adopted here. One may consider training $P(A_k = t|E_k, m_k)$ directly and use it to score paths in the Bell tree. The problem is that 1) the size of I_k from which A_k takes value is variable; 2) the start action depends on all entities in E_k, which makes it difficult to train $P(A_k = t|E_k, m_k)$ directly.

4 Search Issues

As shown in Section 2, the search space of the coreference problem can be represented by the Bell tree. Thus, the search problem reduces to creating the Bell tree while keeping track of path scores and picking the top-N best paths. This is exactly what is described in Algorithm 1.

In Algorithm 1, $\mathcal{H}$ contains all the hypotheses, or paths from the root to the current layer of nodes. Variable $S(E)$ stores the cumulative score for a coreference result E. At line 1, $\mathcal{H}$ is initialized with a single entity consisting of mention m_1, which corresponds to the root node of the Bell tree in Figure 1. Line 2 to 15 loops over the remaining mentions (m_2 to m_n), and for each mention m_k, the algorithm extends each result E in $\mathcal{H}$ (or a path in the Bell tree) by either linking m_k with an existing entity e_i (line 5 to 10), or starting an entity $[m_k]$ (line 11 to 14). The loop from line 2 to 12 corresponds to creating a new layer of nodes for the active mention m_k in the Bell tree. p_m in line 4 and δ in line 6 and 11 have to do with pruning, which will be discussed shortly. The last line returns top N results, where $E_{(r)}$ denotes the r^{th} result ranked by $S(\cdot)$:

$$S(E_{(1)}) \geq S(E_{(2)}) \geq \cdots \geq S(E_{(N)}).$$

Algorithm 1 Search Algorithm

```
Input: mentions $M = \{m_i : 1, \ldots, n\}$;   $N$
Output: top $N$ entity results
1:Initialize: $\mathcal{H} := \{E_1 := \{[m_1]\}\}$ ; $S(E_1) = 1$
2:for $k = 2$ to $n$
3:    foreach node $E \in \mathcal{H}$
4:       compute $p_m$.
5:       foreach $i \in I_k$
6:          if ( $P(L = 1|E, m_k, A = i) > \delta p_m$) {
8:             Extend $E$ to $E'_i$ by linking $m_k$ with $e_i$
9:             $S(E'_i) := S(E) \cdot P(L = 1|E, m_k, A = i)$
10:         }
11:      if($P_\alpha(L = 0|E, m_k) > \delta p_m$) {
12:         Extend $E$ to $E'$ by starting $[m_k]$.
13:         $S(E') := S(E) \cdot P_\alpha(L = 0|E, m_k)$
14:      }
15:   $\mathcal{H} := \{E'\} \cup \{E'_i : i \in I_k\}$.
16:return $\{E_{(1)}, E_{(2)}, \cdots, E_{(N)}\}$
```

The complexity of the search Algorithm 1 is the total number of nodes in the Bell tree, which is $\sum_{k=1}^{n} B(k)$, where $B(k)$ is the Bell Number. Since the Bell number increases rapidly as a function of the number of mentions, pruning is necessary. We prune the search space in the following places:

- Local pruning: any children with a score below a fixed factor δ of the maximum score are pruned. This is done at line 6 and 11 in Algorithm 1. The operation in line 4 is:

  $$p_m := \max\{P_\alpha(L=0|E,m_k)\} \cup \{P(L=1|E,m_k,A=i) : i \in I_k\}.$$

  Block 8-9 is carried out only if $P(L = 1|E,m_k,A=i) > \delta p_m$ and block 12-13 is carried out only if $P_\alpha(L=0|E,m_k) > \delta p_m$.

- Global pruning: similar to local pruning except that this is done using the cumulative score $S(E)$. Pruning based on the global scores is carried out at line 15 of Algorithm 1.

- Limit hypotheses: we set a limit on the maximum number of live paths. This is useful when a document contains many mentions, in which case excessive number of paths may survive local and global pruning.

- Whenever available, we check the compatibility of entity types between the in-focus entity and the active mention. A hypothesis with incompatible entity types is discarded. In the ACE annotation, every mention has an entity type. Therefore we can eliminate hypotheses with two mentions of different types.

5 Experiments

5.1 Performance Metrics

The official performance metric for the ACE task is ACE-value. ACE-value is computed by first calculating the weighted cost of entity insertions, deletions and substitutions; The cost is then normalized against the cost of a nominal coreference system which outputs no entities; The ACE-value is obtained by subtracting the normalized cost from 1. Weights are designed to emphasize NAME entities, while PRONOUN entities (i.e., an entity consisting of only pronominal mentions) carry very low weights. A perfect coreference system will get a 100% ACE-value while a system outputs no entities will get a 0 ACE-value. Thus, the ACE-value can be interpreted as percentage of value a system has, relative to the perfect system.

Since the ACE-value is an entity-level metric and is weighted heavily toward NAME entities, we also measure our system's performance by an entity-constrained mention F-measure (henceforth "ECM-F"). The metric first aligns the system entities with the reference entities so that the number of common mentions is maximized. Each system entity is constrained to align with at most one reference entity, and vice versa. For example, suppose that a reference document contains three entities: $\{[m_1],[m_2,m_3],[m_4]\}$ while a system outputs four entities: $\{[m_1,m_2],[m_3],[m_5],[m_6]\}$, then the best alignment (from reference to system) would be $[m_1] \Leftrightarrow [m_1,m_2]$, $[m_2,m_3] \Leftrightarrow [m_3]$ and other entities are not aligned. The number of common mentions of the best alignment is 2 (i.e., m_1 and m_3), which leads to a mention recall $\frac{2}{4}$ and precision $\frac{2}{5}$. The ECM-F measures the percentage of mentions that are in the "right" entities.

For tests on the MUC data, we report both F-measure using the official MUC score (Vilain et al., 1995) and ECM-F. The MUC score counts the common *links* between the reference and the system output.

5.2 Results on the ACE data

The system is first developed and tested using the ACE data. The ACE coreference system is trained with 416 documents (about $190K$ words) of ACE 2002 training data. A separate 90 documents ($50K$ words) is used as the development-test (Devtest) set. In 2002, NIST released two test sets in February (Feb02) and September (Sep02), respectively. Statistics of the three test sets is summarized in Table 2. We will report coreference results on the true mentions of the three test sets.

TestSet	#-docs	#-words	#-mentions	#-entities
Devtest	90	50426	7470	2891
Feb02	97	52677	7665	3104
Sep02	186	69649	10577	4355

Table 2: Statistics of three test sets.

For the mention-pair model, training events are generated for all compatible mention-pairs, which results in about $909K$ events, about $150K$ of which are positive examples. The full mention-pair model uses about $171K$ features; Most are conjunction features. For the entity-mention model, events are generated by walking through the Bell tree. Only events on the true path (i.e., positive examples) and branches emitting from a node on the true path to a node not on the true path (i.e., negative examples) are generated. For example, in Figure 1, suppose that the path (a)-(b2)-(c4) is the truth, then positive training examples are *starting* event from (a) to (b2) and *linking* event from (b2) to (c4); While the negative examples are *linking* events from (a) to (b1), (b2) to (c3), and the *starting* event from (b2) to (c5). This scheme generates about $322K$ events, out of which about $18K$ are positive training examples. The full entity-mention model has about $8.4K$ features, due to less number of conjunction features and training examples.

Coreference results on the *true* mentions of the Devtest, Feb02, and Sep02 test sets are tabulated in Table 3. These numbers are obtained with a fixed search beam 20 and pruning threshold $\delta = 0.001$ (widening the search beam or using a smaller pruning threshold did not change results significantly).

The mention-pair model in most cases performs better than the mention-entity model by both ACE-value and ECM-F measure although none of the differences is statistically significant (pair-wise t-test) at p-value 0.05. Note that, however, the mention-pair model uses 20 times more features than the entity-pair model. We also observed that, because the score between the in-focus entity and the active mention is computed by (9) in the mention-pair model, the mention-pair sometimes mistakenly places a male pronoun and female pronoun into the same entity, while the same mistake is avoided in the entity-mention model. Using the canonical mentions when computing some features (e.g., “spell”) in the entity-mention model is probably not optimal and it is an area that needs further research.

When the same mention-pair model is used to score the ACE 2003 evaluation data, an ACE-value 73.4% is obtained on the *system*[1] mentions. After retrained with Chinese and Arabic data (much less training data than English), the system got 58.8% and 54.5% ACE-value on the *system* mentions of ACE 2003 evaluation data for Chinese and Arabic, respectively. The results for all three languages are among the top-tier submission systems. Details of the mention detection and coreference system can be found in (Florian et al., 2004).

Since the mention-pair model is better, subsequent analyses are done with the mention pair model only.

5.2.1 Feature Impact

To see how each category of features affects the performance, we start with the aforementioned mention-pair model, incrementally remove each feature category, retrain the system and test it on the Devtest set. The result is summarized in Table 4. The last column lists the number of features. The second row is the full mention-pair model, the third through seventh row correspond to models by removing the syntactic features (i.e., POS tags and apposition features), count features, distance features, mention type and level information, and pair of mention-spelling features. If a basic feature is removed, conjunction features using that basic feature are also removed. It is striking that the smallest system consisting of only 39 features (string and substring match, acronym, edit distance and number of different capitalized words) can get as much as 86.0% ACE-value. Table 4 shows clearly that these lexical features and the distance features are the most important. Sometimes the ACE-value increases after removing a set of features, but the ECM-F measure tracks nicely the trend that the more features there are, the better the performance is. This is because the ACE-value is a weighted metric. A small fluctuation of NAME entities will impact the ACE-value more than many NOMINAL or PRONOUN entities.

[1]System mentions are output from a mention detection system.

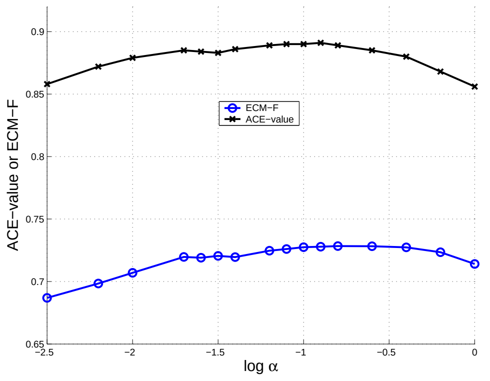

Figure 2: Performance vs. log start penalty

Model	ACE-val(%)	ECM-F(%)	#-features
Full	89.8	73.20 (±2.9)	171K
-syntax	89.0	72.6 (±2.5)	71K
-count	89.4	72.0 (±3.3)	70K
-dist	86.7	*66.2 (±3.9)	24K
-type/level	86.8	65.7 (±2.2)	5.4K
-spell	86.0	64.4 (±1.9)	39

Table 4: Impact of feature categories. Numbers after ± are the standard deviations. * indicates that the result is significantly (pair-wise t-test) different from the line above at $p = 0.05$.

5.2.2 Effect of Start Penalty

As discussed in Section 3.1, the start penalty α can be used to balance the entity miss and false alarm. To see this effect, we decode the Devtest set by varying the start penalty and the result is depicted in Figure 2. The ACE-value and ECM-F track each other fairly well. Both achieve the optimal when $\log\alpha \approx -0.8$.

5.3 Experiments on the MUC data

To see how the proposed algorithm works on the MUC data, we test our algorithm on the MUC6 data. To minimize the change to the coreference system, we first map the MUC data into the ACE style. The original MUC coreference data does not have entity types (i.e., “ORGANIZATION”, “LOCATION” etc), required in the ACE style. Part of entity types can be recovered from the corresponding named-entity annotations. The recovered named-entity label is propagated to all mentions belonging to the same entity. There are 504 out of 2072 mentions of the MUC6 formal test set and 695 out of 2141 mentions of the MUC6 dry-run test set that cannot be assigned labels by this procedure. A

	Devtest		Feb02		Sep02	
Model	ACE-val(%)	ECM-F(%)	ACE-val(%)	ECM-F(%)	ACE-val(%)	ECM-F(%)
MP	89.8	73.2 (±2.9)	90.0	73.1 (±4.0)	88.0	73.1 (±6.8)
EM	89.9	71.7 (±2.4)	88.2	70.8 (±3.9)	87.6	72.4 (±6.2)

Table 3: Coreference results on true mentions: MP – mention-pair model; EM – entity-mention model; ACE-val: ACE-value; ECM-F: Entity-constrained Mention F-measure. MP uses $171K$ features while EM uses only $8.4K$ features. None of the ECM-F differences between MP and EM is statistically significant at $p = 0.05$.

generic type "UNKNOWN" is assigned to these mentions. Mentions that can be found in the named-entity annotation are assumed to have the ACE mention level "NAME"; All other mentions other than English pronouns are assigned the level "NOMINAL."

After the MUC data is mapped into the ACE-style, the same set of feature templates is used to train a coreference system. Two coreference systems are trained on the MUC6 data: one trained with 30 dry-run test documents (henceforth "MUC6-small"); the other trained with 191 "dryrun-train" documents that have both coreference and named-entity annotations (henceforth "MUC6-big") in the latest LDC release.

To use the official MUC scorer, we convert the output of the ACE-style coreference system back into the MUC format. Since MUC does not require entity label and level, the conversion from ACE to MUC is "lossless."

Table 5 tabulates the test results on the *true* mentions of the MUC6 formal test set. The numbers in the table represent the optimal operating point determined by ECM-F. The MUC scorer cannot be used since it inherently favors systems that output fewer number of entities (e.g., putting all mentions of the MUC6 formal test set into one entity will yield a 100% recall and 78.9% precision of links, which gives an 88.2% F-measure). The MUC6-small system compares favorably with the similar experiment in Harabagiu et al. (2001) in which an 81.9% F-measure is reported. When measured by the ECM-F measure, the MUC6-small system has the same level of performance as the ACE system, while the MUC6-big system performs better than the ACE system. The results show that the algorithm works well on the MUC6 data despite some information is lost in the conversion from the MUC format to the ACE format.

System	MUC F-measure	ECM-F
MUC6-small	83.9%	72.1%
MUC6-big	85.7%	76.8%

Table 5: Results on the MUC6 formal test set.

6 Related Work

There exists a large body of literature on the topic of coreference resolution. We will compare this study with some relevant work using machine learning or statistical methods only.

Soon et al. (2001) uses a decision tree model for coreference resolution on the MUC6 and MUC7 data. Leaves of the decision tree are labeled with "link" or "not-link" in training. At test time, the system checks a mention against all its preceding mentions, and the first one labeled with "link" is picked as the antecedent. Their work is later enhanced by (Ng and Cardie, 2002) in several aspects: first, the decision tree returns scores instead of a hard-decision of "link" or "not-link" so that Ng and Cardie (2002) is able to pick the "best" candidate on the left, as opposed the first in (Soon et al., 2001); Second, Ng and Cardie (2002) expands the feature sets of (Soon et al., 2001). The model in (Yang et al., 2003) expands the conditioning scope by including a competing candidate. Neither (Soon et al., 2001) nor (Ng and Cardie, 2002) searches for the global optimal entity in that they make locally independent decisions during search. In contrast, our decoder always searches for the best result ranked by the cumulative score (subject to pruning), and subsequent decisions depend on earlier ones.

Recently, McCallum and Wellner (2003) proposed to use graphical models for computing probabilities of entities. The model is appealing in that it can potentially overcome the limitation of mention-pair model in which dependency among mentions other than the two in question is ignored. However, models in (McCallum and Wellner, 2003) compute directly the probability of an entity configuration conditioned on mentions, and it is not clear how the models can be factored to do the incremental search, as it is impractical to enumerate all possible entities even for documents with a moderate number of mentions. The Bell tree representation proposed in this paper, however, provides us with a naturally incremental framework for coreference resolution.

Maximum entropy method has been used in coreference resolution before. For example, Kehler (1997) uses a mention-pair maximum entropy model, and two methods are proposed to compute entity scores based on the mention-pair model: 1) a distribution over entity space is deduced; 2) the most recent mention of an entity, together with the candidate mention, is used to compute the entity-mention score. In contrast, in our mention pair model, an entity-mention pair is scored by taking the maximum score among possible mention pairs. Our entity-mention model eliminates the need to synthesize an entity-mention score from mention-pair scores. Morton (2000) also uses a maximum entropy mention-pair model, and a special "dummy" mention is used to model the event of starting a new entity. Features involving the dummy mention are essentially computed with the single (normal) mention, and therefore the starting model is weak. In our model, the starting model is obtained by "complementing" the linking scores. The advantage is that we do not need to train a starting model. To compensate the model inaccuracy, we introduce a "starting penalty" to balance the linking and starting scores.

To our knowledge, the paper is the first time the Bell tree is used to represent the search space of the coreference resolution problem.

7 Conclusion

We propose to use the Bell tree to represent the process of forming entities from mentions. The Bell tree represents the search space of the coreference resolution problem. We studied two maximum entropy models, namely the mention-pair model and the entity-mention model, both of which can be used to score entity hypotheses. A beam search algorithm is used to search the best entity result. State-of-the-art performance has been achieved on the ACE coreference data across three languages.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred. We also would like to thank the anonymous reviewers for suggestions of improving the paper.

References

E.T. Bell. 1934. Exponential numbers. *Amer. Math. Monthly*, pages 411–419.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March.

R Florian, H Hassan, A Ittycheriah, H Jing, N Kambhatla, X Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 1–8, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proc. of the sixth Workshop on Very Large Corpora*.

Sanda M. Harabagiu, Razvan C. Bunescu, and Steven J. Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proc. of NAACL*.

J. Hobbs. 1976. Pronoun resolution. Technical report, Dept. of Computer Science, CUNY, Technical Report TR76-1.

A. Ittycheriah, L. Lita, N. Kambhatla, N. Nicolov, S. Roukos, and M. Stys. 2003. Identifying and tracking entity mentions in a maximum entropy framework. In *HLT-NAACL 2003: Short Papers*, May 27 - June 1.

Andrew Kehler. 1997. Probabilistic coreference in information extraction. In *Proc. of EMNLP*.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Procs. of the 17th Internaltional Conference on Computational Linguistics*, pages 869–875.

Thomas S. Morton. 2000. Coreference for NLP applications. In *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference(MUC-6)*, San Francisco, CA. Morgan Kaufmann.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*, pages 104–111.

NIST. 2003. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

Adwait Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 1 – 10.

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

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, , and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *In Proc. of MUC6*, pages 45–52.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proc. of the* 41^{st} *ACL*.