Multi-Lingual Coreference Resolution With Syntactic Features

Xiaoqiang Luo and Imed Zitouni
1101 Kitchawan Road
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.
{xiaoluo, izitouni}@us.ibm.com

Abstract

In this paper, we study the impact of a group of features extracted automatically from machine-generated parse trees on coreference resolution. One focus is on designing syntactic features using the binding theory as the guideline to improve pronoun resolution, although linguistic phenomenon such as apposition is also modeled. These features are applied to the Arabic, Chinese and English coreference resolution systems and their effectiveness is evaluated on data from the Automatic Content Extraction (ACE) task. The syntactic features improve the Arabic and English systems significantly, but play a limited role in the Chinese one. Detailed analyses are done to understand the syntactic features' impact on the three coreference systems.

1 Introduction

A coreference resolution system aims to group together *mention*s referring to the same *entity*, where 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*. In the following example:

(I) "John believes himself to be the best student."

mentions are underlined. The three mentions "John", "himself", "the best student" are of type name, pronoun [1], and nominal, respectively. They form an *entity* since they all refer to the same person.

Syntactic information plays an important role in coreference resolution. For example, the binding theory (Haegeman, 1994; Beatrice and Kroch, 2000) provides a good account of the constraints on the antecedent of English pronouns. The theory relies on syntactic parse trees to determine the governing category which defines the scope of binding constraints. We will use the theory as a guideline to help us design features in a machine learning framework.

Previous pronoun resolution work (Hobbs, 1976; Lappin and Leass, 1994; Ge et al., 1998; Stuckardt, 2001) explicitly utilized syntactic information before. But there are unique challenges in this study: (1) Syntactic information is extracted from parse trees automatically generated. This is possible because of the availability of statistical parsers, which can be trained on human-annotated treebanks (Marcus et al., 1993; Xia et al., 2000; Maamouri and Bies, 2004) for multiple languages; (2) The binding theory is used as a guideline and syntactic structures are encoded as features in a maximum entropy coreference system; (3) The syntactic features are evaluated on three languages: Arabic, Chinese and English (one goal is to see if features motivated by the English language can help coreference resolution in other languages). All contrastive experiments are done on publicly-available data; (4) Our coreference system resolves coreferential relationships among all the annotated mentions, not just for pronouns.

Using machine-generated parse trees eliminates the need of hand-labeled trees in a coreference system. However, it is a major challenge to extract useful information from these noisy parse trees. Our approach is encoding the structures contained in a parse tree into a set of computable features, each of which is associated with a weight automatically determined by a machine learning algorithm. This contrasts with the approach of extracting rules and assigning weights to these rules by hand (Lappin and Leass, 1994; Stuckardt, 2001). The advantage of our approach is robustness: if a particular structure is helpful, it will be assigned a high weight; if a feature is extracted from a highly noisy parse tree and is not informative in coreference resolution, it will be assigned a small weight. By avoiding writing rules, we automatically incorporate useful information into our model and at the same time limit the potentially negative impact from noisy parsing output.

---

[1]"Pronoun" in this paper refers to both anaphor and normal pronoun.

2 Statistical Coreference Resolution Model

Our coreference system uses a binary entity-mention model $P_L(\cdot|e, m)$ (henceforth "link model") to score the action of linking a mention m to an entity e. In our implementation, the link model is computed as

$$P_L(L = 1|e, m) \approx \max_{m' \in e} \hat{P}_L(L = 1|e, m', m), \quad (1)$$

where m' is one mention in entity e, and the basic model building block $\hat{P}_L(L = 1|e, m', m)$ is an exponential or maximum entropy model (Berger et al., 1996):

$$\hat{P}_L(L|e, m', m) = \frac{\exp\left\{\sum_i \lambda_i g_i(e, m', m, L)\right\}}{Z(e, m', m)}, \quad (2)$$

where $Z(e, m', m)$ is a normalizing factor to ensure that $\hat{P}_L(\cdot|e, m', m)$ is a probability, $\{g_i(e, m', m, L)\}$ are features and $\{\lambda_i\}$ are feature weights.

Another *start* model is used to score the action of creating a new entity with the current mention m. Since starting a new entity depends on all the partial entities created in the history $\{e_i\}_{i=1}^t$, we use the following approximation:

$$P_S(S = 1|e_1, e_2, \cdots, e_t, m) \approx$$
$$1 - \max_{1 \le i \le t} P_L(L = 1|e_i, m) \quad (3)$$

In the maximum-entropy model (2), feature (typically binary) functions $\{g_i(e, m', m, \cdot)\}$ provide us with a flexible framework to encode useful information into the the system: it can be as simple as "$g_i(e, m', m, L = 1) = 1$ if m' and m have the same surface string," or "$g_j(e, m', m, L = 0) = 1$ if e and m differ in number," or as complex as "$g_l(e, m', m, L = 1) = 1$ if m' c-commands m and m' is a NAME mention and m is a pronoun mention." These feature functions bear similarity to rules used in other coreference systems (Lappin and Leass, 1994; Mitkov, 1998; Stuckardt, 2001), except that the feature weights $\{\lambda_i\}$ are automatically trained over a corpus with coreference information. Learning feature weights automatically eliminates the need of manually assigning the weights or precedence of rules, and opens the door for us to explore rich features extracted from parse trees, which is discussed in the next section.

3 Syntactic Features

In this section, we present a set of features extracted from syntactic parse trees. We discuss how we *approximately* compute linguistic concepts such as governing category (Haegeman, 1994), apposition and dependency relationships from noisy syntactic parse trees. While parsing and parse trees depend on the target language, the automatic nature of feature extraction from parse trees makes the process language-independent.
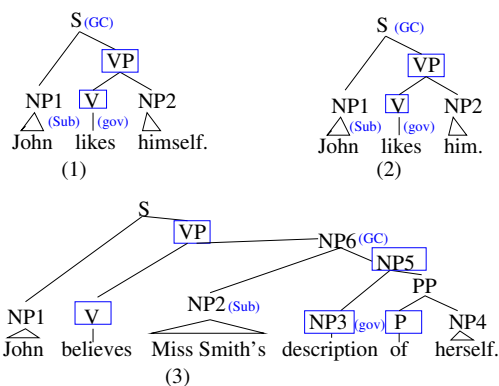

Figure 1: GC examples.

3.1 Features Inspired by Binding Theory

The binding theory (Haegeman, 1994) concerning pronouns can be summarized with the following principles:

1. A reflexive or reciprocal pronoun (e.g., "herself" or "each other") must be bound in its governing category (GC).
2. A normal pronoun must be free in its governing category.

The first principle states that the antecedent of a reflexive or reciprocal pronoun is within its GC, while the second principle says that the antecedent of a normal pronoun is outside its GC. While the two principles are simple, they all rely on the concept of governing category, which is defined as the minimal domain containing the pronoun in question, its governor, and an accessible subject.

The concept GC can best be explained with a few examples in Figure 1, where the label of a head constituent is marked within a box, and GC, accessible subject, and governor constituents are marked in parentheses with "GC", "Sub" and "gov." Noun-phrases (NP) are numbered for the convenience of referencing. For example, in sub-figure (1) of Figure 1, the governor of "himself" is "likes," the subject is "John," hence the GC is the entire sentence spanned by the root "S." Since "himself" is reflexive, its antecedent must be "John" by Principle 1. The parse tree in sub-figure (2) is the same as that in sub-figure (1), but since "him" is a normal pronoun, its antecedent, according to Principle 2, has to be outside the GC, that is, "him" cannot be coreferenced with "John." Sentence in sub-figure (3) is slightly more complicated: the governor of "herself" is "description," and the accessible subject is "Miss Smith." Thus, the governing category is NP6. The first principle implies that the antecedent of "herself" must be "Miss Smith."

It is clear from these examples that GC is very useful in finding the antecedent of a pronoun. But the last example shows that determining GC is not a trivial matter. Not only is the correct parse tree required, but extra information is also needed to identify the head governor

and the minimal constituent dominating the pronoun, its governor and an accessible subject. Determining the accessible subject itself entails checking other constraints such as number and gender agreement. The complexity of computing governing category, compounded with the noisy nature of machine-generated parse tree, prompts us to compute a set of features that characterize the structural relationship between a candidate mention and a pronoun, as opposed to explicitly identify GC in a parse tree. These features are designed to implicitly model the binding constraints.

Given a candidate antecedent or mention m_1 and a pronoun mention m_2 within a parsed sentence, we first test if they have c-command relation, and then a set of counting features are computed. The features are detailed as follows:

(1) C-command $ccmd(m_1, m_2)$: A constituent X c-commands another constituent Y in a parse tree if the first branching node dominating X also dominates Y. The binary feature $ccmd(m_1, m_2)$ is true if the minimum NP dominating m_1 c-commands the minimum NP dominating m_2. In sub-figure (1) of Figure 1, NP1 c-commands NP2 since the first branching node dominating NP1 is S and it dominates NP2.

If $ccmd(m_1, m_2)$ is true, we then define the c-command path $T(m_1, m_2)$ as the path from the minimum NP dominating m_2 to the first branching node that dominates the minimum NP dominating m_1. In sub-figure (1) of Figure 1, the c-command path $T(\text{"John", "himself"})$ would be "NP2-VP-S."

(2) $NP\_count(m_1, m_2)$: If $ccmd(m_1, m_2)$ is true, then $NP\_count(m_1, m_2)$ counts how many NPs are seen on the c-command path $T(m_1, m_2)$, excluding two endpoints. In sub-figure (1) of Figure 1, $NP\_count(\text{"John", "himself"}) = 0$ since there is no NP on $T(\text{"John", "himself"})$.

(3) $VP\_count(m_1, m_2)$: similar to $NP\_count(m_1, m_2)$, except that this feature counts how many verb phrases (VP) are seen on the c-command path. In sub-figure (1) of Figure 1, $VP\_count(\text{"John", "himself"})$ is true since there is one VP on $T(\text{"John", "himself"})$.

(4) $S\_count(m_1, m_2)$: This feature counts how many clauses are seen on the c-command path when $ccmd(m_1, m_2)$ is true. In sub-figure (1) of Figure 1, $S\_count(\text{"John", "himself"}) = 0$ since there is no clause label on $T(\text{"John", "himself"})$.

These features are designed to capture information in the concept of governing category when used in conjunction with attributes (e.g., gender, number, reflexiveness) of individual pronouns. Counting the intermediate NPs, VPs and sub-clauses implicitly characterizes the governor of a pronoun in question; the presence or absence of a sub-clause indicates whethere or not a coreferential relation is across clause boundary.

3.2 Dependency Features

In addition to features inspired by the binding theory, a set of dependency features are also computed with the help of syntactic parse trees. This is motivated by examples such as "John is the president of ABC Corporation," where "John" and "the president" refer to the same person and should be in the same entity. In scenarios like this, lexical features do not help, while the knowledge that "John" left-modifies the verb "is" and the "the president" right-modifies the same verb would be useful.

Given two mentions m_1 and m_2 in a sentence, we compute the following dependency features:

(1) $same\_head(m_1, m_2)$: The feature compares the bi-lexical dependencies $\langle m_1, h(m_1) \rangle$, and $\langle m_2, h(m_2) \rangle$, where $h(x)$ is the head word which x modifies. The feature is active only if $h(m_1) = h(m_2)$, in which case it returns $h(m_1)$.

(2) $same\_POS(m_1, m_2)$: To get good coverage of dependencies, we compute a feature $same\_POS(m_1, m_2)$, which examines the same dependency as in (1) and returns the common head part-of-speech (POS) tag if $h(m_1) = h(m_2)$.

The head child nodes are marked with boxes in Figure 1. For the parse tree in sub-figure (1), $same\_head(\text{"John", "him"})$ would return "likes" as "John" left-modifies "likes" while "him" right-modifies "likes," and $same\_POS(\text{"John", "him"})$ would return "V" as the POS tag of "likes" is "V."

(3) $mod(m_1, m_2)$: the binary feature is true if m_1 modifies m_2. For parse tree (2) of Figure 1, $mod(\text{"John", "him"})$ returns false as "John" does not modify "him" directly. A reverse order feature $mod(m_2, m_1)$ is computed too.

(4) $same\_head2(m_1, m_2)$: this set of features examine second-level dependency. It compares the head word of $h(m_1)$, or $h(h(m_1))$, with $h(m_2)$ and returns the common head if $h(h(m_1)) = h(m_2)$. A reverse order feature $same\_head2(m_2, m_1)$ is also computed.

(5) $same\_POS2(m_1, m_2)$: similar to (4), except that it computes the second-level POS. A reverse order feature $same\_POS2(m_2, m_1)$ is computed too.

(6) $same\_head22(m_1, m_2)$: it returns the common second-level head if $h(h(m_1)) = h(h(m_2))$.

3.3 Apposition and Same-Parent Features

Apposition is a phenomenon where two adjacent NPs refer to the same entity, as "Jimmy Carter" and "the former president" in the following example:

(II) "Jimmy Carter, the former president of US, is visiting Europe."

Note that not all NPs separated by a comma are necessarily appositive. For example, in "John called Al, Bob, and Charlie last night," "Al" and "Bob" share a same NP

parent and are separated by comma, but they are not appositive.

To compute the apposition feature $appos(m_1, m_2)$ for mention-pair (m_1, m_2), we first determine the minimum dominating NP of m_1 and m_2. The minimum dominating NP of a mention is the lowest NP, with an optional modifying phrase or clause, that spans the mention. If the two minimum dominating NPs have the same parent NP, and they are the only two NP children of the parent, the value of $appos(m_1, m_2)$ is true. This would exclude "Al" and "Bob" in "John called Al, Bob, and Charlie last night" from being computed as apposition.

We also implement a feature $same\_parent(m_1, m_2)$ which tests if two mentions m_1 and m_2 are dominated by a common NP. The feature helps to prevent the system from linking "his" with "colleague" in the sentence "John called his colleague."

All the features described in Section 3.1-3.3 are computed from syntactic trees generated by a parser. While the parser is language dependent, feature computation boils down to encoding the structural relationship of two mentions, which is language independent. To test the effectiveness of the syntactic features, we integrate them into 3 coreference systems processing Arabic, Chinese and English.

4 Experimental Results

4.1 Data and System Description

All experiments are done on true mentions of the ACE (NIST, 2004) 2004 data. We reserve part of LDC-released 2004 data as the development-test set (henceforth "devtest") as follows: documents are sorted by their date and time within each data source (e.g., broadcast news (bnews) and news wire (nwire) are two different sources) and the last 25% documents of each data source are reserved as the devtest set. Splitting data on chronological order simulates the process of a system's development and deployment in the real world. The devtest set statistics of three languages (Arabic, Chinese and English) is summarized in Table 1, where the number of documents, mentions and entities is shown on row 2 through 4, respectively. The rest of 2004 ACE data together with earlier ACE data is used as training.

	Arabic	Chinese	English
#-docs	178	166	114
#-mentions	11358	8524	7008
#-entities	4428	3876	2929

Table 1: Devtest Set Statistics by Language

The official 2004 evaluation test set is used as the blind test set on which we run our system once after the system development is finished. We will report summary results on this test set.

As for parser, we train three off-shelf maximum-entropy parsers (Ratnaparkhi, 1999) using the Arabic, Chinese and English Penn treebank (Maamouri and Bies, 2004; Xia et al., 2000; Marcus et al., 1993). Arabic words are segmented while the Chinese parser is a character-based parser. The three parsers have a label F-measure of 77%, 80%, and 86% on their respective test sets. The three parsers are used to parse both ACE training and test data. Features described in Section 3 are computed from machine-generated parse trees.

Apart from features extracted from parse trees, our coreference system also utilizes other features such as lexical features (e.g., string matching), distance features characterized as quantized word and sentence distances, mention- and entity-level attribute information (e.g, ACE distinguishes 4 types of mentions: NAM(e), NOM(inal), PRE(modifier) and PRO(noun)) found in the 2004 ACE data. Details of these features can be found in (Luo et al., 2004).

4.2 Performance Metrics

The official performance metric in the ACE task is ACE-Value (NIST, 2004). The ACE-Value is an entity-based metric computed by subtracting a normalized cost from 1 (so it is unbounded below). The cost of a system is a weighted sum of costs associated with entity misses, false alarms and errors. This cost is normalized against the cost of a nominal system that outputs no entity. A perfect coreference system gets 100% ACE-Value while a system outputting many false-alarm entities could get a negative value.

The default weights in ACE-Value emphasize names, and severely discount pronouns: the relative importance of a pronoun is two orders of magnitude less than that of a name. So the ACE-Value will not be able to accurately reflect a system's improvement on pronouns[2]. For this reason, we compute an *unweighted* entity-constrained mention F-measure (Luo, 2005) and report all contrastive experiments with this metric. The F-measure is computed by first aligning system and reference entities such that the number of common mentions is maximized and 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: $\{[m_1, m_2], [m_3], [m_5], [m_6]\}$, where $\{m_i : i = 1, 2, \cdots, 6\}$ are mentions, 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

[2]Another possible choice is the MUC F-measure (Vilain et al., 1995). But the metric has a systematic bias for systems generating fewer entities (Bagga and Baldwin, 1998) – see Luo (2005). Another reason is that it cannot score single-mention entity.

(i.e., m_1 and m_3), thus the recall is $\frac{2}{4}$ and precision is $\frac{2}{5}$. Due to the one-to-one entity alignment constraint, the F-measure here is more stringent than the accuracy (Ge et al., 1998; Mitkov, 1998; Kehler et al., 2004) computed on antecedent-pronoun pairs.

4.3 Effect of Syntactic Features

We first present the contrastive experimental results on the devtest described in sub-section 4.1.

Two coreference systems are trained for each language: a baseline without syntactic features, and a system including the syntactic features. The entity-constrained F-measures with mention-type breakdown are presented in Table 2. Rows marked with N_m contain the number of mentions, while rows with "base" and "+synt" are F-measures for the baseline and the system with the syntactic features, respectively.

The syntactic features improve pronoun mentions across three languages – not surprising since features inspired by the binding theory are designed to improve pronouns. The pronoun improvement on the Arabic (from 73.2% to 74.6%) and English (from 69.2% to 72.0%) system is statistically significant (at above 95% confidence level), but change on the Chinese system is not. For Arabic, the syntactic features improve Arabic NAM, NOM and PRE mentions, probably because Arabic pronouns are sometimes attached to other types of mentions. For Chinese and English, the syntactic features do not practically change the systems' performance.

As will be shown in Section 4.5, the baseline systems without syntactic features are already competitive, compared with the results on the coreference evaluation track (EDR-coref) of the ACE 2004 evaluation (NIS, 2004). So it is nice to see that syntactic features further improve a good baseline on Arabic and English.

Arabic					
	Mention Type				
	NAM	NOM	PRE	PRO	Total
N_m	2843	3438	1291	3786	11358
base	86.8	73.2	86.7	73.2	78.2
+synt	**88.4**	**76.4**	**87.4**	**74.6**	**80.1**
Chinese					
N_m	4034	3696	-	794	8524
base	**95.4**	**77.8**	-	65.9	**85.0**
+synt	95.2	77.7	-	**66.5**	84.9
English					
N_m	2069	2173	835	1931	7008
base	92.0	73.4	**88.7**	69.2	79.6
+synt	92.0	**75.3**	87.8	**72.0**	**80.8**

Table 2: F-measure(%) Breakdown by Mention Type: NAM(e), NOM(inal), PRE(modifier) and PRO(noun). Chinese data does not have the PRE type.

4.4 Error Analyses

From the results in Table 2, we know that the set of syntactic features are working in the Arabic and English system. But the results also raise some questions: Are there interactions among the the syntactic features and other features? Why do the syntactic features work well for Arabic and English, but not Chinese? To answer these questions, we look into each system and report our findings in the following sections.

4.4.1 English System

Our system uses a group of distance features. One observation is that information provided by some syntactic features (e.g., $VP\_count(m_1, m_2)$ etc) may have overlapped with some of the distance features. To test if this is the case, we take out the distance features from the English system, and then train two systems, one with the syntactic features, one without. The results are shown in Table 3, where numbers on the row "b-dist" are F-measures after removing the distance features from the baseline, and numbers on the row "b-dist+synt" are with the syntactic features.

	Mention Type				
	NAM	NOM	PRE	PRO	Total
b-dist	84.2	68.8	74.6	63.3	72.5
b-dist+synt	**90.7**	**74.2**	**87.8**	**69.0**	**79.3**

Table 3: Impact of Syntactic Features on English System After Taking out Distance Features. Numbers are F-measures(%).

As can be seen, the impact of the syntactic features is much larger when the distance features are absent in the system: performance improves across all the four mention types after adding the syntactic features, and the overall F-measure jumps from 72.5% to 79.3%. The PRE type gets the biggest improvement since features extracted from parse trees include apposition, same-parent test, and dependency features, which are designed to help mention pairs in close distance, just as in the case of PRE mentions.

Comparing the numbers in Table 3 with the English baseline of Table 2, we can also conclude that distance features and syntactic features lead to about the same level of performance when the other set of features is not used. When the distance features are used, the syntactic features further help to improve the performance of the NOM and PRO mention type, albeit to a less degree because of information overlap between the two sets of features.

4.4.2 Chinese System

Results in Table 2 show that the syntactic features are not so effective for Chinese as for Arabic and English. The

first thing we look into is if there is any idiosyncrasy in the Chinese language.

In Table 4, we list the statistics collected over the training sets of the three languages: the second row are the total number of mentions, the third row the number of pronoun mentions, the fourth row the number of events where the c-command feature $ccmd(m_1, m_2)$ is used, and the last row the average number of c-command features per pronoun (i.e., the fourth row divided by the third row). A pronouns event is defined as a tuple of training instance (e, m_1, m_2) where m_1 is a mention in entity e, and the second mention m_2 is a pronoun.

From Table 4, it is clear that Chinese pronoun distribution is very different: pronoun mentions account for about 8.7% of the total mentions in Chinese, while 29.0% of Arabic mentions and 25.1% of English mentions are pronouns (the same disparity can be observed in the devtest set in Table 2). This is because Chinese is a pro-drop language (Huang, 1984): for example, in the Chinese Penn treebank version 4, there are 4933 overt pronouns, but 5750 pro-drops! The ubiquity of pro-drops in Chinese results in signigicantly less pronoun training events. Consequently, the pronoun-related features are not trained as well as in English and Arabic. One way to quantify this is by looking at the average number of c-command features on a per-pronoun basis: as shown in the last row of Table 4, the c-command feature is seen more than twice often in Arabic and English as in Chinese. Since low-count features are filtered out, the sparsity of pronoun events prevent many compound features (e.g., conjunction of syntactic and distance features) from being trained in the Chinese system, which explains why the syntactic features do not help Chinese pronouns.

	Arabic	Chinese	English
#total-mentions	31706	33851	58202
#pron-mentions	9183	2941	14635
#-ccmd-event	10236	1260	13691
#ccmd/pron	1.14	0.428	0.936

Table 4: Distribution of Pronoun Mentions and Frequency of c-command Features

4.4.3 Arabic System

As stated in Table 4, 29.0% of Arabic mentions are pronouns, compared to a slightly lower number (25.1%) for English. This explains the relatively high positive impact of the syntactic features on the Arabic coreference system, compared to English and Chinese systems. To understand how syntactic features work in the Arabic system, we examine two examples extracted from the devtest set: (1) the first example shows the negative impact of syntactic features because of the noisy parsing output, and (2) the second example proves the effectiveness of the syntactic features to find the dependency between two

mentions. In both examples, the baseline system and the system with syntactic features give different results.

Let's consider the following sentence:

··· وتعتبر إسرَئيل **القدس** عَاصمتهَا ···

... its-capital ← Jerusalem ← Israel ← consider ← and ...

فيمَا يريد الفلسطينيون الشطر الشرقي **للمدينة** ···

of-the-city ← the-Eastern ← the-half ← the-Palestininan ← want ← while

The English text shown above is a word-to-word translation of the Arabic text (read from right-to-left). In this example, the parser wrongly put the nominal mention الــقــدس (Jerusalem) and the pronominal mention المدينة (the-city) under the same constituent, which activates the *same_parent* feature. The use of the feature *same_parent*(المدينة, القدس) leads to the two mentions being put into different entities. This is because there are many cases in the training data where two mentions under the same parent are indeed in different entities: a similar English example is "*John called his sister*", where "*his*" and "*sister*" belong to two different entities. The *same_parent* feature is a strong indicator of not putting them into the same entity.

كَان + **ال** + **زقَاقيون** + ي + حَاول + ون + نهب + ال + محل + ات + ال + تجَاري + ة ب + حجة + أَن + **هم** + ···
kAn + **Al** + **zqAqywn** + y + HAwl + wn + nhb + Al + mHl + At + Al + tjAry + p + b + Hjp + An + **hm** + ...
was + **the** + **zqAqywn** + *present-verb-marker* y + trying + *plural-verb-marker* wn + to-steal + the + office + s + the + commercial + s + with + excuse + that + **they** + ...

Table 5: An example where syntactic features help to link the PRO mention هم (hm) with its antecedent, the NAM mention الزقَاقيون (AlzqAqywn): top – Arabic sentence; middle – corresponding romanized sentence; bottom – token-to-token English translation.

Table 5 shows another example in the devtest set. The top part presents the segmented Arabic text, the middle part is the corresponding romanized text, and the bottom part contains the token-to-token English translation. Note that Arabic text reads from right to left and its corresponding romanized text from left to right (i.e., the right-most Arabic token maps to the left-most romanized token). The parser output the correct syntactic structure: Figure 2 shows a portion of the system-generated parse tree. It can be checked that NP1 c-commands NP2 and the group of features inspired by the binding theory are active. These features help to link the PRO(onominal) mention هم (hm) with the NAM(e) mention الزقَاقيون (AlzqAqywn). Without syntactic features theses two mentions were split into different entities.
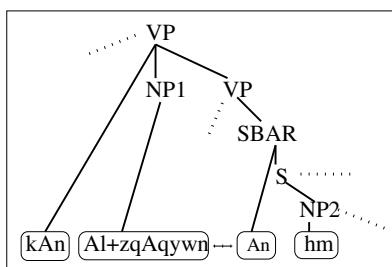
Figure 2: A Portion of the Syntactic Tree.

4.5 ACE 2004 Results

To get a sense of the performance level of our system, we report the results on the ACE 2004 official test set with both the F-measure and the official ACE-Value metric. This data is used as the blind test set which we run our system only once.

Results are summarized in Table 6, where the second row (i.e. "base") contains the baseline numbers, and the third row (i.e., "+synt") contains the numbers from systems with the syntactic features. Columns under "F" are F-measure and those under "AV" are ACE-Value. The last row N_m contains the number of mentions in the three test sets.

	Arabic		Chinese		English	
	F	AV	F	AV	F	AV
base	80.1	88.0	84.7	92.7	80.6	90.9
+synt	81.5	88.9	84.7	92.8	82.0	91.6
N_m	11358		11178		10336	

Table 6: Summary Results on the 2004 ACE Evaluation Data.

The performance of three full ("+synt") systems is remarkably close to that on the devtest set(cf. Table 2): For Arabic, F-measure is 80.1 on the devtest vs. 81.5 here; For Chinese, 84.9 vs. 84.7; And for English, 80.8 vs. 82.0. The syntactic features again help Arabic and English – statistically very significant in F-measure, but have no significant impact on Chinese. The performance consistency across the devtest and blind test set indicates that the systems are well trained.

The F-measures are computed on all types of mentions. Improvement on mention-types targeted by the syntactic features is larger than the lump-sum F-measure. For example, the F-measure for English pronouns on this test set is improved from 69.5% to 73.7% (not shown in Table 6 due to space limit). The main purpose of Table 6 is to get a sense of performance level correspondence between the F-measure and ACE-Value.

Also note that, for Arabic and English, the difference between the "base" and "+synt" systems, when measured by ACE-Value, is much smaller. This is not surprising since ACE-Value heavily discounts pronouns and is in-

sensitive to improvement on pronouns – the very reason we adopt the F-measure in Section 4.3 and 4.4 when reporting the contrastive experiment results.

5 Related Work

Many researchers have used the syntactic information in their coreference system before. For example, Hobbs (1976) uses a set of rules that are applied to parse trees to determine the antecedent of a pronoun. The rule precedence is determined heuristically and no weight is used. Lappin and Leass (1994) extracted rules from the output of the English Slot Grammar (ESG) (McCord, 1993). Rule weights are assigned manually and the system resolves the third person pronouns and reflexive pronouns only. Ge et al. (1998) uses a non-parametrized statistical model to find the antecedent from a list of candidates generated by applying the Hobbs algorithm to the English Penn Treebank. Kehler et al. (2004) experiments making use of predicate-argument structure extracted from a large TDT-corpus. Compared with these work, our work uses machine-generated parse trees from which trainable features are extracted in a maximum-entropy coreference system, while (Ge et al., 1998) assumes that correct parse trees are given. Feature weights are automatically trained in our system while (Lappin and Leass, 1994; Stuckardt, 2001) assign weights manually.

There are a large amount of published work (Morton, 2000; Soon et al., 2001; Ng and Cardie, 2002; Yang et al., 2003; Luo et al., 2004; Kehler et al., 2004) using machine-learning techniques in coreference resolution. But none of these work tried to compute complex linguistic concept such as governing category [3] . Our work demonstrates how relevant linguistic knowledge can be derived automatically from system-generated parse trees and encoded into computable and trainable features in a machine-learning framework.

6 Conclusions

In this paper, linguistic knowledge is used to guide us to design features in maximum-entropy-based coreference resolution systems. In particular, we show how to compute a set of features to approximate the linguistic notions such as governing category and apposition, and how to compute the dependency features using syntactic parse trees. While the features are motivated by examining English data, we see significant improvements on both English and Arabic systems. Due to the language idiosyncrasy (e.g., pro-drops), we do not see the syntactic features change the Chinese system significantly.

---

[3]Ng and Cardie (2002) used a BINDING feature, but it is not clear from their paper how the feature was computed and what its impact was on their system.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation (LREC'98)*, pages 563–566.

Santorini Beatrice and Anthony Kroch. 2000. *The syntax of natural language: An online introduction using the Trees program*. www.ling.upenn.edu/ beatrice/syntax-textbook.

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.

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

Liliane Haegeman. 1994. *Introduction to Government and Binding Theory*. Basil Blackwell Inc., 2nd edition.

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

C.-T. James Huang. 1984. On the distribution and reference of empty pronouns. *Linguistic Inquiry*, 15:531–574.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. The (Non)utility of predicate-argument frequencies for pronoun interpretation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4), December.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. of ACL*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Procs. of HLT/EMNLP*.

Mohamed Maamouri and Ann Bies. 2004. Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, COLING*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Michael McCord. 1993. Heuristics for broad-coverage natural language parsing. In *Proc. ARPA Human Language Technology Workshop*.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Procs. of the 36th ACL/17th COLING*, 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*.

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

NIST. 2004. *Proceedings of ACE Evaluation and PI Meeting 2004 Workshop*, Alexandria, VA, September.

NIST. 2004. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–178.

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.

Roland Stuckardt. 2001. Design and enhanced evaluation of a robust anaphor resolution algorithm. *Computational Linguistics*, 27(4).

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.

F. Xia, M. Palmer, N. Xue, M.E. Okurowski, J. Kovarik, F.D. Chiou, S. Huang, T. Kroch, and M. Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proc of the 2nd Intl. Conf. on Language Resources and Evaluation (LREC 2000)*.

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