Corpus-Based Learning for Noun Phrase Coreference Resolution

Wee Meng Soon
Hwee Tou Ng
Chung Yong Lim
DSO National Laboratories
20 Science Park Drive, Singapore 118230
{sweemeng,nhweetou,lchungyo}@dso.org.sg

Abstract

In this paper, we present a learning approach for coreference resolution of noun phrases in unrestricted text. The approach learns from a small, annotated corpus and the task includes resolving not just pronouns but rather general noun phrases. In contrast to previous work, we attempt to evaluate our approach on a common data set, the MUC-6 coreference corpus. We obtained encouraging results, indicating that on the general noun phrase coreference task, the learning approach holds promise and achieves accuracy comparable to non-learning approaches.

1 Introduction

Coreference resolution refers to the process of determining if two expressions in natural language refer to the same entity in the world. It is an important subtask in natural language processing systems. In particular, information extraction (IE) systems like those built in the DARPA Message Understanding Conferences (Chinchor, 1998; Sundheim, 1995) have revealed that coreference resolution is such a critical component of IE systems that a separate coreference subtask has been defined and evaluated since MUC-6 (Committee, 1995).

In this paper, we focus on the task of determining coreference relations as defined in MUC-6 (Committee, 1995). Specifically, a coreference relation denotes an identity of reference and holds between two textual elements known as *markables*, which are nouns, noun phrases, or pronouns. Thus, our coreference task resolves general noun phrases and not just pronouns, unlike in some previous work on anaphora resolution. The ability to link co-referring noun phrases both within and across sentences is critical to discourse analysis and language understanding in general.

2 A Learning Approach for Coreference Resolution

We adopt a corpus-based, learning approach for noun phrase coreference resolution. In this approach, we need a relatively small corpus of training documents that have been annotated with coreference chains of noun phrases. All possible markables in a training document are determined by a pipeline of language processing modules, and training examples in the form of feature vectors are generated for appropriate pairs of markables. These training examples are then given to a learning algorithm to build a classifier. To determine the coreference chains in a new document, all markables are determined and potential pairs of co-referring markables are presented to the classifier which will decide whether the two markables actually co-refer. We give the details of these steps in the following subsections.

2.1 Determination of Markables

A pre-requisite for coreference resolution is to obtain most, if not all, of the possible markables in a raw input text. To determine the markables, a pipeline of natural language processing (NLP) modules is used. They consist of sentence segmentation, tokenization, morphological analysis, part-of-speech tagging, noun phrase identification, named entity recognition, and semantic class determination. As far as coreference resolution is concerned, the goal of these NLP modules is to determine the boundary of the markables, and to provide the necessary information about each markable for subsequent generation of features in the training examples.

Our part-of-speech tagger is a standard statistical bigram tagger based on the Hidden Markov Model (HMM) (Church, 1988). Similarly, we built a statistical HMM-based noun phrase identification module where the noun phrase boundaries are determined solely based on the part-of-speech tags assigned to the words in a sentence. We also implemented a module that recognizes MUC-style named entities, i.e., organization, person, location, date, time, money, and percent. Our named entity recognition module uses the HMM approach of (Bikel et al., 1999; Bikel et al., 1997), which learns from a tagged corpus of named entities. That is, our part-of-speech tagger, noun phrase identification module, and the named entity recognition module are all based on HMM and learn from corpora tagged with parts-of-speech, noun phrases, and named entities, respectively. The markables needed for coreference resolution is the union of the noun phrases and named entities found.

To achieve high accuracy in coreference resolution, it is most critical that the eligible candidates for coreference are identified correctly in the first place. In order to test the effectiveness of our system in determining the markables, we attempted to match the markables generated by our system against those appearing in the coreference chains annotated in 100 SGML documents, a subset of the documents available in MUC-6. We found that our system is able to correctly identify about 85% of the noun phrases appearing in coreference chains in the 100 annotated SGML documents. Most of the unmatched noun phrases are of the following types: (1) Our system generated a head noun which is a subset of the noun phrase in the annotated corpus. For example, "Saudi Arabia, the cartel's biggest producer," was annotated as a markable but our system generated only "Saudi Arabia". (2) Our system extracted a sequence of words that cannot be considered as a markable. (3) Unclear notion of what constitutes a markable. For example, "wage reductions" was annotated, but "selective wage reductions" was identified by our system instead.

2.2 Determination of Feature Vectors

Feature vectors are required for training and testing the coreference engine. A feature vector consists of 10 features described below, and is derived based on two extracted markables, i and j, where i is the antecedent and j is the anaphor. Information needed to derive the feature vectors is provided by the pipeline of language modules prior to the coreference engine.

1. **Distance Feature** Its possible values are 0, 1, 2, 3, ... This feature captures the distance between i and j. If i and j are in the same sentence, the value is 0; if they are 1 sentence apart, the value is 1; and so on.

2. **Pronoun Feature** Its possible values are true or false. If j is a pronoun, return true; else return false. Pronouns include reflexive pronouns (himself, herself), personal pronouns (he, him, you), and possessive pronouns (hers, her).

3. **String Match Feature** Its possible values are true or false. If the string of i matches the string of j, return true; else return false.

4. **Definite Noun Phrase Feature** Its possible values are true or false. In our definition, a definite noun phrase is a noun phrase that starts with the word "the". For example, "the car" is a definite noun phrase. If j is a definite noun phrase, return true; else return false.

5. **Demonstrative Noun Phrase Feature** Its possible values are true or false. A demonstrative noun phrase is one that starts with the word "this", "that", "these" or "those". If j is a demonstrative noun phrase, then return true; else return false.

6. **Number Agreement Feature** Its possible values are true or false. If i and j agree in number, i.e., they are both singular or both plural, the value is true; otherwise false. Pronouns such as "they", "them", etc., are plural, while "it", "him", etc., are singular. The morphological root of a noun is used to determine whether it is singular or plural if the noun is not a pronoun.

7. **Semantic Class Agreement Feature** Its possible values are true, false, or unknown. In our system, we defined the following semantic classes: "female", "male", "person", "organization", "location", "date", "time", "money", "percent", and "object". These semantic classes are

arranged in a simple ISA hierarchy. Each of the "female" and "male" semantic classes is a subclass of the semantic class "person", while each of the semantic classes "organization", "location", "date", "time", "money", and "percent" is a subclass of the semantic class "object". Each of these defined semantic classes is then mapped to a WORDNET synset (Miller, 1990). For example, "male" is mapped to sense 2 of the noun "male" in WORDNET, "location" is mapped to sense 1 of the noun "location", etc.

The semantic class determination module assumes that the semantic class for every markable extracted is the first sense of the head noun of the markable. Since WORDNET orders the senses of a noun by their frequency, this is equivalent to choosing the most frequent sense as the semantic class for each noun. If the selected semantic class of a markable is a subclass of one of our defined semantic class C, then the semantic class of the markable is C, else its semantic class is "unknown".

The semantic classes of markables i and j are in agreement if one is the parent of the other (e.g., "chairman" with semantic class "person" and "Mr. Lim" with semantic class "male"), or both of them are the same (e.g., "Mr. Lim" and "he" both of semantic class "male"). The value returned for such cases is true. If the semantic classes of i and j are not the same (e.g. "IBM" with semantic class "organization" and "Mr. Lim" with semantic class "male"), return false. If either semantic class is "unknown", then the head nouns of both markables are compared. If they are the same, return true, else return unknown.

8. **Gender Agreement Feature** Its possible values are true, false, or unknown. The gender of a markable is determined in several ways. Designators and pronouns such as "Mr.", "Mrs.", "she", "he", etc., can determine the gender. For a markable that is a person's name such as "Peter H. Diller", the gender cannot be determined by the above method. In our system, the gender of such a markable can only be determined if there are markables found later in the document that refer to "Peter H. Diller" by using the designator-form of the name, such as "Mr. Diller". The gender of a markable will be unknown for noun phrases such as "the president", "chief executive officer", etc. If the gender of either markable i or j is unknown, then the gender agreement feature value is unknown; else if i and j agree in gender, then the feature value is true; otherwise its value is false.

9. **Proper Name Feature** Its possible values are true or false. A proper name is determined based on capitalization. Prepositions appearing in the name such as "of", "and", etc., need not be in upper case. If i and j are both proper names, return true; else return false.

10. **Alias Feature** Its possible values are true or false. If i is an alias of j or vice versa, return true; else return false. That is, this feature value is true if i and j are proper names that refer to the same entity. For example, the pairs "Mr. Simpson" and "Bent Simpson", "IBM" and "International Business Machines Corp.", "SEC" and "the Securities and Exchange Commission", "Mr. Dingell" and "Chairman John Dingell", are aliases. However, the pairs "Mrs. Washington" and "her", "the talk" and "the meeting", are not aliases.

2.3 Generating Training Examples

Consider a coreference chain A1 - A2 - A3 - A4 found in an annotated training document. Only pairs of noun phrases in the chain that are immediately adjacent (i.e., A1 - A2, A2 - A3, and A3 - A4) are used to generate the positive training examples. The first noun phrase in a pair is always considered the antecedent while the second is the anaphor. On the other hand, negative training examples are extracted as follows. For each antecedent-anaphor pair, first obtain all markables between the antecedent and the anaphor. These markables are either not found in any coreference chain or they appear in other chains. Each of them is then paired with the anaphor to form a negative example. For example, if markables a, b, B1 appear between A1 and A2, then the negative examples are a - A2, b - A2 and B1 - A2. Note that a and b do not appear in any coreference chain while B1 appears in another coreference chain.

For an annotated noun phrase in a coreference chain in a training document, the same noun phrase must be identified as a markable by our pipeline of language processing modules before this noun phrase can be used to form a feature vector for use as a training example. This is because the information necessary to derive a feature vector, such as semantic class and gender, is computed by the language modules. If an annotated noun phrase is not identified as a markable, it will not contribute any training example. Note that the language modules are also needed to identify markables not already annotated in the training document so that they can used for generating the negative examples.

2.4 Building a Classifier

The next step is to use a machine learning algorithm to learn a classifier based on the feature vectors generated from the training documents. The learning algorithm used in our coreference engine is C4.5 (Quinlan, 1993). C4.5 is a commonly used machine learning algorithm and thus it may be considered as a baseline method against which other learning algorithms can be compared.

2.5 Generating Coreference Chains for Test Documents

Before determining the coreference chains for a test document, all possible markables need to be extracted from the document. Every markable is a possible anaphor, and every markable before the anaphor in document order is a possible antecedent of the anaphor, except when the anaphor is nested. If the anaphor is a child or nested markable, then its possible antecedents must not be any markable with the same root markable as the current anaphor. However, the possible antecedents can be other root markables and their children that are before the anaphor in document order. For example, consider the 2 root markables, "Mr. Tom's daughter" and "His daughter's eyes", appearing in that order in a test document. The possible antecedents of "His" cannot be "His daughter", nor "His daughter's eyes", but can be "Mr. Tom" or "Mr. Tom's daughter".

The coreference resolution algorithm considers every markable j starting from the second markable in the document to be a potential candidate as an anaphor. For each j, the algorithm considers every markable before j as a potential antecedent. For each pair i and j, a feature vector is generated and given to the decision tree classifier. A co-referring antecedent is found if the classifier returns true. The algorithm starts from the immediately preceding markable and proceeds backwards in the reverse order of the markables in the document until there is no more markable to test or an antecedent is found.

3 Evaluation

In order to evaluate the performance of our learning approach to coreference resolution on a common data set, we utilized the annotated corpus and scoring program from MUC-6, which assembled a set of newswire documents annotated with coreference chains. Although we did not participate in MUC-6, we were able to obtain the MUC-6 training and test corpus from the MUC organizers for research purpose.[1] 30 dry-run documents annotated with coreference information were used as the training documents for our coreference engine. After training the engine, we tested its accuracy on the 30 formal test documents in MUC-6. These 30 test documents are exactly those used to evaluate the systems that participated in the MUC-6 evaluation.

Our implemented system runs on a Pentium II 400MHz PC. The total size of the 30 training documents is close to 13,000 words. It took less than five minutes to generate the training examples from these training documents. The training time for the C4.5 algorithm to generate a decision tree from all the training examples was about 30 seconds. The decision tree classifier learned (using a pruning confidence level of 25%) is shown in Figure 1.

One advantage of using a decision tree learning algorithm is that the resulting decision tree classifier built can be interpreted by human. The decision tree in Figure 1 seems to encapsulate a reasonable rule-of-thumb that matches our intuitive linguistic notion of when two noun phrases can co-refer. It is also interesting to note that only five out of the ten available features in the training examples are actually used in the final decision tree built.

[1]Contact Beth Sundheim at sundheim@nosc.mil

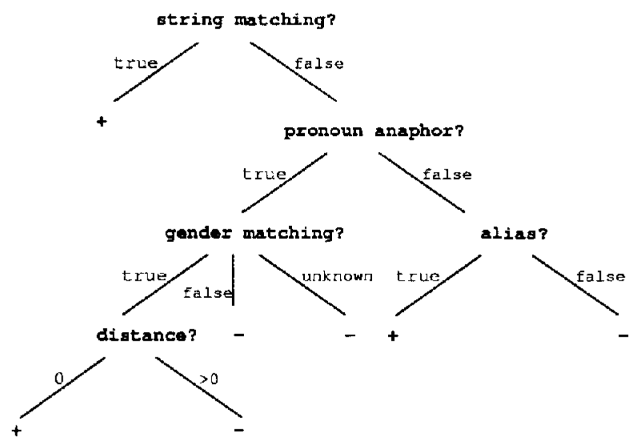

Figure 1: The decision tree classifier learned

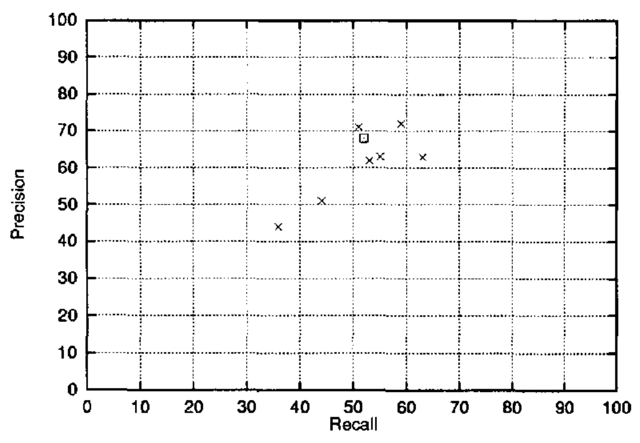

Figure 2: Coreference scores of MUC-6 systems and our system

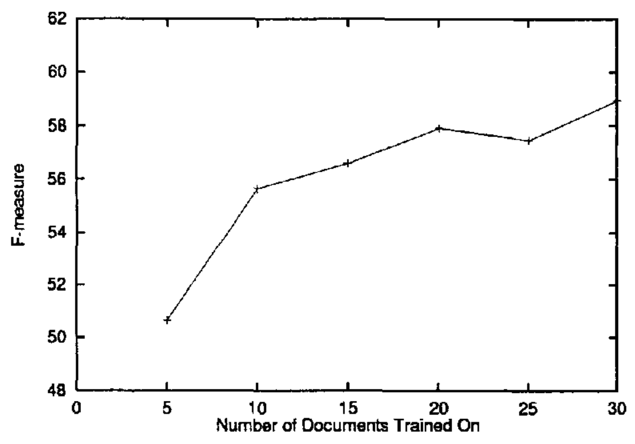

Figure 3: Learning curve of coreference resolution accuracy

When given new test documents, the output of the coreference engine is in the form of SGML files with the coreference chains properly annotated according to the MUC-6 guidelines. The time taken to generate the coreference chains for 30 test documents of close to 14,000 words was less than three minutes. We then used the scorer program of MUC-6 to generate the recall and precision score for our coreference engine.

Our coreference engine achieves a recall of 52% and a precision of 68%, yielding a balanced F-measure of 58.9%. We plotted the score of our coreference engine (square-shaped) against the other official test scores of MUC-6 systems (cross-shaped) in Figure 2. We also plotted the learning curve of our coreference engine in Figure 3, showing its accuracy averaged over five random trials when trained on 5, 10, ..., 30 training documents.

Our score is in the upper region of the MUC-6 systems. We performed a simple two-tailed, paired t-test at $p = 0.05$ to determine whether the difference between our system's F-measure scores and each of the other MUC-6 systems' F-measure scores on the 30 formal test documents is statistically significant. We found that at the 95% significance level, our system performed worse than one, better than two, and as well as the rest of the MUC-6 systems. Our result is encouraging as it indicates that a learning approach using relatively shallow features and a small number of training documents can lead to scores that are comparable to systems built using non-learning approaches.

It should be noted that the accuracy of our coreference resolution engine depends to a large extent on the performance of the NLP modules that are executed before the coreference engine. Our current learning-based, HMM named entity recognition module is trained on 318 documents (a disjoint set from the 30 formal test documents) tagged with named entities, and its score on the MUC-6 named entity task for the 30 formal test documents is only 88.9%, which is not considered very high by MUC-6 standard. For example, our named entity recognizer could not identify the two named entities "USAir" and "Piedmont" in the expression "USAir and Piedmont" but instead treat it as one single named entity. Also, some of the features such as number agreement, gender agreement and semantic class agreement are difficult to determine at times. For example, "they" is sometimes used to refer to "the government" even though superficially both do not seem to agree in number. All these problems hurt the performance of the coreference engine.

4 Related Work

There is a long tradition of work on coreference resolution within computational linguistics, but most of them are not subjected to empirical evaluation until recently. Among the work that reported quantitative evaluation results, most are not based on learning from an annotated corpus (Baldwin, 1997; Kameyama, 1997; Lappin and Leass, 1994; Mitkov, 1997).

To our knowledge, the work of (Aone and Bennett, 1995; Ge et al., 1998; McCarthy and Lehnert, 1995) are the only ones that are based on learning from an annotated corpus. Ge et al. (Ge et al., 1998) used a statistical model for resolving pronouns. In contrast, we used a decision tree learning algorithm and resolved general noun phrases, not just pronouns. Both the work of (Aone and Bennett, 1995; McCarthy and Lehnert, 1995) employed decision tree learning. However, the features they used include domain-specific ones like DNP-F (definite NP whose referent is a facility) (Aone and Bennett, 1995), JV-CHILD-i (does i refer to a joint venture formed as the result of a tie-up) (McCarthy and Lehnert, 1995), etc. In contrast, all our 10 features are domain-independent, which makes our coreference engine a domain-independent module. Moreover, both (Aone and Bennett, 1995) and (McCarthy and Lehnert, 1995) made simplifying assumptions in their experimental evaluations. Since the accuracy of coreference resolution relies on the correct identification of the candidate noun phrases, both (Aone and Bennett, 1995) and (McCarthy and Lehnert, 1995) only evaluated their systems on noun phrases that have been correctly identified. In contrast, we evaluated our coreference resolution engine as part of a total system which has to first identify all the candidate noun phrases and has to deal with the inevitable noisy data when mistakes occur in noun phrase identification. Also, the evaluation of (Aone and Bennett, 1995) and (McCarthy and Lehnert, 1995) only focused on specific types of noun phrases (organizations and business entities), and (Aone and Bennett, 1995) dealt only with Japanese texts. Our evaluation was done on all types of English noun phrases instead.

None of the systems in MUC-7 adopted a learning approach to coreference resolution (Chinchor, 1998). Among the MUC-6 systems, the only one that we can directly compare to is the UMass system, which also used C4.5 for coreference resolution. The other MUC-6 systems were not based on a learning approach. The score of the UMass system is not high compared to the rest of the MUC-6 systems. In particular, the system's recall is relatively low. As explained in (Fisher et al., 1995), the reason for this is that it only concentrated on coreference relationships among references to people and organizations. Our system, as opposed to the UMass system, considered all types of markables. The score of our system is higher than that of the UMass system, and the difference is statistically significant at $p = 0.05$. Thus, the contribution of our work is in showing that a learning approach, when evaluated on a common coreference data set, is able to achieve accuracy competitive with state-of-the-art systems using non-learning approaches.

5 Conclusion

In this paper, we presented a learning approach for coreference resolution of noun phrases in unrestricted text. The approach learns from a small, annotated corpus and the task includes resolving not just pronouns but rather general noun phrases. In contrast to previous work, we evaluated our approach on a common data set, the MUC-6 coreference corpus. We obtained encouraging results, indicating that on the general noun phrase coreference task, the learning approach holds promise and achieves accuracy comparable to non-learning approaches.

6 Acknowledgements

We would like to thank Beth Sundheim who made available to us the MUC-6 data set without which this work would have been impossible. We would also like to thank Hai Leong Chieu who implemented the HMM-based named entity learning module.

References

Chinatsu Aone and Scott William Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.*

Breck Baldwin. 1997. CogNIAC: High precision coreference with limited knowledge and linguistic resources. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 38–45.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3), February.

Nancy A. Chinchor. 1998. Overview of MUC-7/MET-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7) http://www.muc.saic.com/proceedings/muc_7_toc.html.*

Kenneth Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

MUC-6 Program Committee. 1995. Coreference task definition (v2.3, 8 Sep 95). In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 335–344.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 127–140.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170.

Megumi Kameyama. 1997. Recognizing referential links: An information extraction perspective. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 46–53.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, December.

Joseph F. McCarthy and Wendy Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1050–1055.

George A. Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Ruslan Mitkov. 1997. Factors in anaphora resolution: They are not the only things that matter. a case study based on two different approaches. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 14–21.

John Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Francisco, CA.

Beth M. Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 13–31.