A statistical coreference solver

George Ytterbrink
D01, Lund Institute of Technology, Sweden
d01gl@efd.lth.se

Abstract

This paper presents a learning-based approach to coreference resolution of noun phrases. The system learns from a small, annotated corpus to solve coreference between general noun phrases by extracting feature vectors that are used by Weka to build a classifier.

1 Introduction

Coreference resolution is the process of determining whether two expressions in natural language refer to the same entity in the world. A coreference relation denotes an identity of reference and holds between two textual elements known as markables, which can be definite noun phrases, demonstrative noun phrases, proper names, appositives, sub-noun phrases that act as modifiers, pronouns, and so on. Determining coreference relations is an important subtask in natural language processing systems, especially in information extraction systems. This paper describes a way to solve the task of determining coreference relations between general noun phrases.

2 Determination of feature vectors

To build a learning-based coreference engine, a set of features that is useful in determining whether two markables corefer or not is needed. The features are extracted from an annotated corpus.
The feature vector consists of a total of six features described below. The features are derived based on two extracted markables, i and j, where i is the potential antecedent and j the anaphor.

String match (STR_MATCH): Articles (a, an, the) and demonstrative pronouns (this, these, that, those) are removed from the strings before performing the comparison. If the string of i matches the string of j, 1 is returned, else 0 is returned. E.g. *the document* matches *this document.*

Distance (DIST): This feature is the distance between i and j counted in sentences. If i and j are in the same sentence, 0 is returned, if they are one sentence apart, 1 is returned, and so on.

i-pronoun (I_PROUNOUN): If i is a pronoun, 1 is returned, else 0 is returned. Pronouns include reflexive (himself, herself), personal (he, him, you) and possessive pronouns (hers, her).

j-pronoun (J_PRONOUN): If j is a pronoun (as described above), 1 is returned, else 0 is returned.

Definite noun phrase (DEF_NP): A definite noun phrase is a noun phrase that starts with the word *the.* Eg. *the document* is a definite noun phrase. If j is a definite noun phrase, 1 is returned, else 0 is returned.

Demonstrative noun phrase (DEM_NP): A demonstrative noun phrase is a noun phrase that starts with one of the words *this, that, these* or *those.* If j is a demonstrative noun phrase, 1 is returned, else 0 is returned.

As an example 2 shows the feature vector for the antecedant *i, Linux* and the anaphor *j, it*, in the following sentence:

The Linux Access HOWTO covers the use of adaptive technology with Linux, in particular, using adaptive technology to make *Linux* accessible to those who could not use *it* otherwise.

Feature	Value	Comment
STR_MATCH	0	i and j do not match
DIST	0	i and j are in the same sentence
I_PRONOUN	0	i is not a pronoun
J_PRONOUN	1	j is a pronoun
DEF_NP	0	j is not a definite noun phrase
DEM_NP	0	j is not a demonstrative noun phrase

3 Building a classifier

Training examples are extracted from an annotated training document. In the training document there are coreference chains e.g. A1 - A2 - A3 - A4. The noun phrases in the chain that are immediately adjacent are used in pairs to generate the positive training examples (i.e. A1 - A2, A2 - A3, A3 - A4). The first noun phrase in a pair is always considered the antecedent, while the second is the anaphor. Between the two members of each antecedent-anaphor pair, there are other noun phrases that either are not found in any coreference chain or appear in other chains. Each of them is then paired with the anaphor to form a negative example. For example, if markables a, b, and B1 appear between A1 and A2, then the negative examples are a - A2, b - A2, and B1 - A2.
An example of how training examples are extracted:

(The Linux Access HOWTO)$_{a1}$ covers (the use of (adaptive technology)$_{b1}$)$_{c1}$ with (Linux)$_{a2}$, in particular, using (adaptive technology)$_{b2}$ to make (Linux)$_{a3}$ accessible to (those who could not use (it)$_{a4}$)$_{d1}$ otherwise.

Looking at the chain *a*, which is about Linux. There are four noun phrases that corefer: (The Linux Access HOWTO)$_{a1}$ matches with (Linux)$_{a2}$, (Linux)$_{a2}$ with (Linux)$_{a3}$ and (Linux)$_{a3}$ with (it)$_{a4}$. The positive training examples that are extracted are: ((The Linux Access HOWTO)$_{a1}$, (Linux)$_{a2}$), ((Linux)$_{a2}$, (Linux)$_{a3}$) and ((Linux)$_{a3}$, (it)$_{a4}$).
The noun phrases that are between (The Linux Access HOWTO)$_{a1}$ and (Linux)$_{a2}$ are used to generate the negative training examples. The negative examples are: ((adaptive technology)$_{b1}$, (Linux)$_{a2}$) and (((the use of adaptive technology)$_{c1}$), (Linux)$_{a2}$). Negative Examples can be found in the same way between ((Linux)$_{a2}$, (Linux)$_{a3}$) and ((Linux)$_{a3}$, (it)$_{a4}$).
The training examples are used with Weka to build a classifier. Weka was set to use the j48 decision tree classifier.

4 Conclusions

Given the time allocated for the project there was no time to implement a evaluation for the decision tree. Because of that the only results that can be presented are the statistics from weka. The classifier achieved a recall of 30.2% and a precision of 78%, giving a F-measure of 43.5%. Soon et al achieves a recall of 58.6% and a precision of 67.3%, giving a F-measure of 62.6% (using MUC-6 evaluation).

The generated decision tree classifier for the Linux Access HOWTO:

$STR\_MATCH = 0 : 0$
$STR\_MATCH = 1 : 1$

Only the string matching gave any information that could be used. One reason for this could be that the annotated corpus that was used is a technical document, in which very few pronouns occur and words are often repeated. It would be interesting to see how it would have performed on a corpus consisting of dialogues instead.

Other features that would have been interesting to look at are:

- Gender, decide how the gender of i and j match.
- Alias, if i is an alias to j or vice versa, i.e. if i and j are named entities (person, date, organization, etc.) that refer to the same entity.
- Appositive, if j is in apposition to i, e.g.

the markable *the chairman of Microsoft Corp* is in apposition to *Bill Gates* in the sentence *Bill Gates, the chairman of Microsoft Corp ....*

- Number Agreement, if i and j agree in number, i.e. they are both singular or both plural.

- Both Proper Name, if i and j are both proper names.

Most of these features require more information than was available in the used corpus and was therefor not implemented. Further work on this classifier would be to decide if any of these or other features should be added, a program that can give more correct evaluation would also be needed.

References

Wee Meng Soon, Daniel Chung Yong Lim and Hwee Tou Ng. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases.

Weka, http://www.cs.waikato.ac.nz/ml/weka/

Coreferentially annotated corpora, http://clg.wlv.ac.uk/resources/corefann.php

MUC-6, http://cs.nyu.edu/cs/faculty/grishman/muc6.html