ELERFED: Final Report

Massimo Poesio, David Day
Ron Artstein, Jason Duncan, Vladimir Eidelman,
Claudio Giuliano, Rob Hall, Janet Hitzeman,
Alan Jern, Mijail Kabadjov, Stanley Yong Wai Keong,
Gideon Mann, Alessandro Moschitti, Simone Ponzetto,
Jason Smith, Josef Steinberger, Michael Strube,
Jian Su, Yannick Versley, Xiaofeng Yang and Michael Wick

October 16, 2007

Contents

Chapter 1

Introduction and General Overview

Massimo Poesio

1.1 Introduction

The goal of the ELERFED 2007 Johns Hopkins Workshop was to explore the contribution of lexical and encyclopedic knowledge to three different versions of the **Entity Disambiguation** task.

The term entity disambiguation refers to the task of identifying which **mentions** of entities in documents refer to the same object, and which instead refer to different ones. The term **intra-document coreference**, or IDC, will be used to indicate entity disambiguation limited to mentions occurring in the same document only. This task is also called simply 'coreference' or 'entity tracking' when only links between mentions realized with proper names are established (as in *Prime Minister Gordon Brown* ... *Mr. Brown*) and 'anaphora resolution' when all anaphoric mentions are considered (as in *Prime Minister Gordon Brown* ... *he*).[1] By contrast, the term **cross-document coreference**, or CDC will be used to indicate the task of identifying coreference across documents. Both IDC and CDC can be considered clustering problems (Cardie and Wagstaff 1999), in which the goal is to cluster mentions forming so-called **coreference chains**. A simplified form of CDC is the so-called **web entity** task, which is based on the assumption that each document is only about one person, and where documents instead

[1] There is considerable theoretical discussion concerning the proper characterization of this task: see, e.g., (van Deemter and Kibble 2000; Poesio 2004a).

of mentions are clustered. (Instances of this task are the **web people** task at SEMEVAL (Artiles et al. 2007) and the Spock Challenge.)

The motivation for this workshop was the growing evidence that large corpora such as the Web and community-built repositories of knowledge such as Wikipedia may help us solve the single biggest problem in entity disambiguation, the need for large amounts of lexical and encyclopedic knowledge to track entities. For instance, lexical knowledge–that *proposal* and *plan* are quasi-synonyms–is necessary to identify *the Packwood proposal* as the antecedent for *the Packwood plan* in the following example.

(1a1) **The Packwood proposal** would reduce the tax depending on how long an asset was held. It also would create a new IRA that would shield from taxation the appreciation on investments made for a wide variety of purposes, including retirement, medical expenses, first-home purchases and tuition.

b. A White House spokesman said President Bush is "generally supportive" of the Packwood plan.

Whereas in the following example, knowing that *The FCC* is an agency is necessary to choose between that possible antecedent and the other most likely candidate, *AT&T*.

(1a2) **The FCC** took three specific actions regarding [AT&T]. By a 4-0 vote, it allowed AT&T to continue offering special discount packages to big customers, called Tariff 12, rejecting appeals by AT&T competitors that the discounts were illegal.

b. The agency said that because MCI's offer had expired AT&T couldn't continue to offer its discount plan.

Early work in NLP suggested that hand-coding such knowledge wouldn't scale up; the result had been abandoning the hope to achieve high performance on the coreference resolution task. However, more recently, methods for automatically extracting features encoding such knowledge from corpora (Poesio et al. 2004; Markert and Nissim 2005; Versley 2007) and Wikipedia (Ponzetto and Strube 2006) have been shown to lead to improved results: for instance, the knowledge that the FCC is an agency, needed to interpret (1.2) correctly, is all contained in the first paragraph of the Wikipedia entry for the FCC. Work such as (Bunescu and Pasca 2006) suggests that lexical and encyclopedic knowledge may also lead to improved results at the cross-document coreference and web people tasks.

In the rest of this introductory chapter we first briefly summarize the state of the art in Entity Disambiguation, then we summarize the research carried out during the workshop and its preliminary results, in the order in which the work is then presented in more detail in the rest of the report. We first report on our work on Web People and Cross-Document Coreference. We then discuss our work on Intra Document Coreference, divided in three parts: a discussion of the BART IDC architecture, followed by a discussion of our research on statistical models of coreference; after which we discuss our work on extracting lexical and encyclopedic knowledge. In the end, we discuss our work on evaluating entity disambiguation, in particular our efforts towards creating the annotated corpora used for such evaluation.

1.2 State of the art in Entity Disambiguation

1.2.1 Intra-document coreference

Research in intra-document coreference, or anaphora resolution, has been carried out since the seventies (Charniak 1972; Hobbs 1978b; Sidner 1979; Carter 1987; Hobbs et al. 1993; Lappin and Leass 1994b) but large-scale empirical investigations and the development of systems able to process large amounts of data have only began fairly recently, particularly after the creation of annotated resources as part of the Message Understanding and ACE initiatives (Aone and Bennett 1995b; Kehler 1997; Poesio and Vieira 1998a; Cardie and Wagstaff 1999; Vieira and Poesio 2000; Soon et al. 2001b; Ng and Cardie 2002b; McCallum and Wellner 2004; Yang et al. 2004b; Ponzetto and Strube 2006; Culotta et al. 2007). These initiatives also led to the development of novel evaluation methods, the best known among which is the model theoretical approach proposed by Vilain et al. (1995).

The model proposed by Soon et al. (2001b), a fully automatic system attempting to resolve all types of nominal anaphora, has become the standard baseline against which work in this area is evaluated; we made this choice as well. The Soon et al system is based on a very simple model of the anaphora resolution task as a binary classification task in which ⟨anaphor, antecedent⟩ pairs are classified as standing in a coreference relation or not, on the basis of 12 features encoding string-based, agreement, and distance information; a single feature encodes semantic class agreement. Soon et al also proposed methods for generating training instances and for choosing a candidate which have also since become fairly standard. The system was evaluated on the MUC-6 and MUC-7 corpora, achieving for MUC-6 a recall of 58.6%, a precision of 67.3%, and an F-measure of 62.6%, whereas for MUC-7, it achieved a recall of 56.1%, a precision of 65.5%, and an F measure of 60.4%.

Even more recently, the first usable tools for intra-doc coreference started to appear, such as GUITAR (Poesio and Kabadjov 2004). Even though the performance of such systems is limited, as one would expect given the figures reported above, nevertheless it has already been shown that even such limited performance may result to significant improvements in performance in tasks such as summarization (Steinberger et al. 2007) and relation extraction. As relation extraction is important for both cross-document coreference and web people, this suggests that intra-document coreference may contribute indirectly to these tasks, as well.

An analysis of the errors produced by the Soon *et al.* method indicates three main sources of problems.

1. Mention identification. Except when working with a very limited number of types of easily identifiable mentions, these systems tend to miss a great many mentions.

2. Overly simplified model of the coreference task, particularly of the process by which antecedents are chosen.

3. Excessive reliance on head string matching.

4. Lack of lexical and commonsense knowledge.

The example in (1.3) illustrates the problems with such systems. Our reimplementation of the Soon et al algorithm does not identify any of the coreference links relating mentions of Petrie Stores; on the other end, because head string matching overrides every other factor, and because information about postmodification is not used, *the first six months of fiscal 1994* is identified as the antecedent of *the first six months of fiscal 1993.*

(1a3) [Petrie Stores Corporation, Secaucus, NJ,] said an uncertain economy and faltering sales probably will result in a second quarter loss and perhaps a deficit for the first six months of fiscal 1994.

b. [The womenâĂŹs appareil specialty retailer] said sales at stores open more than one year, a key barometer of a retain concern strength, declined 2.5% in May, June and the first week of July.

c. [The company] operates 1714 stores.

d. In the first six months of fiscal 1993, [the company] had net income of $1.5 million âĂę.

Work in the last five years has aimed at improving the Soon et al model. Improved models of mention identification viewing the task as a case of joint inference have been proposed, e.g., by Daume and Marcu (2005). More sophisticated models of the coreference task have been proposed by Ng and Cardie (2002b); McCallum and Wellner (2004); Yang et al. (2004b); Daume and Marcu (2005); Culotta et al. (2007). Using kernels, Yang et al. (2006a) were able to incorporate a more sophisticated treatment of syntactic features handling some types of binding constraints. Finally, there have been some promising attempts at using using lexical and encyclopedic knowledge. We discuss this work next.

1.2.2 Using lexical and encyclopedic knowledge for large-scale IDC

There are two main strands of research on using lexical and commonsense knowledge to help coreference on unrestricted text: one devoted to the use of information about the semantic structure of verbs and primarily concerned with improving precision in pronoun resolution; and a second one concerned with the use of hyponymy information to improve recall in the resolution of nominals. In the workshop we focused on the second strand of work.

information to (Harabagiu and Moldovan 1998; Poesio et al. 1997) generally involved using WordNet (Fellbaum 1998). The problem with WordNet, especially in those days, was poor recall, both for synonymy and for hyponymy: e.g., Poesio et al. (1997) found a recall of about 30% for hyponymy. Poesio et al. (1998) attempted to replace synonymy in WordNet with semantic similarity computed in an unsupervised fashion from corpora, whereas Poesio et al. (2002) used patterns (to extract information about meronymy), but the corpus used (the British National Corpus) did not provide enough recall. The next key development was using patterns over the Web (Markert and Nissim 2005; Poesio et al. 2004): this gave reasonable recall, particularly in combination with WordNet. Finally, Ponzetto and Strube (2006) showed that Wikipedia, again in combination with WordNet, also contained enough information in its category structure to lead to significant improvements in performance. Versley (2007) systematically investigated the relative strengths and weaknesses of all sources of knowledge considered above and their combinations (for German).

It is worth pointing out that all of these efforts were focused on increasing recall for nominals; we are not aware of any effort to use information so extracted to increase precision by exploiting information about modifiers.

1.2.3 Cross Document coreference and Web People

Work on cross-document coreference began more recently than work on IDC (Bagga and Baldwin 1998a), but there has been much development in recent years (Mann and Yarowsky 2003; Blume 2005; Bunescu and Pasca 2006; Chen and Martin 2007) because of great interest both from government and from industry (as shown, e.g., the creation of Spock[2] and the Spock challenge). In particular there has been great interest in a simpler form of entity disambiguation, generally known as Web entity as in the case of the Web people task of Semeval (Artiles et al. 2007).

As testified by the SEMEVAL Web People task (Artiles et al. 2007), most state of the art systems are based on unsupervised clustering of entity descriptions containing a mixture of collocational and other information, among which information about entities and relations. SEMEVAL also showed that the clustering technique and especially the termination criterion are crucial.

Just as in the case of IDC, prior to this year this area suffered from a lack of data; no sizeable dataset existed until the creation of the SEMEVAL and then of the Spock datasets, neither of which however is entirely satisfactory.

1.3 Web People and Cross-Document Coreference

1.3.1 Web People

Both Web People and CDC are naturally viewed as a clustering problem. Our research in this area focused therefore on several types of clustering algorithms, preferrably ones that would work with all types of entity disambiguation.

The first method we studied is a discriminative model, similar to that used by Culotta et al. (2007) for intra-doc coreference–a first order model using features over sets of mentions. In the version used for Web People, a maximum entropy approach is used to estimate the probability $p(y_i|x^i)$, where $y_i = true$ if and only if all documents in x^i refer to the same underlying entity. The Metropolis-Hastings method was used to modify the solutions proposed by a standard greedy agglomerative clustering algorithm. A third novelty was the use, in addition to the usual features (bags of words, n-grams, named entities), of an unsupervised model of lexical knowledge, Latent Dirichlet Allocation (Blei et al. 2003), that can find several topics for each document. In our experiments with a subset of the Spock dataset,

[2] `www.spock.com`

Metropolis-Hastings was found to outperform simple greedy agglomerative, and using topics as features led to further improvements.

The great effectiveness of topic models with the discriminative model suggested testing a generative model where the implicit variables modelled topics, and using Gibbs sampling to perform inference. We also tested an extension of the basic model in which certain words have more importance than others.

Our work on the Web People task is discussed in some detail in Chapter 2.

1.3.2 CDC

Although the CDC task is more complex than Web People, the ACE 2005 CDC corpus proved highly unambiguous, resulting in a very high baseline: assuming that all mentions with the same name corefer results in a B-Cubed value of .80. Using the discriminative model developed for the Web People task, and all the features, thus achieved an extremely high B-Cubed value of .96.

1.3.3 Relation extraction (Jian Su)

Relation extraction modules to be used for both IDC and CDC have been developed by the I2R team (Su Jian, Stanley Yong), Claudio Giuliano from FBK-IRST and Gideon Mann from Uni Amherst. Although we did not have enough time to run tests using this information for Web People, CDC and IDC, we plan to do so in the near future.

I2R

I2R trained both a supervised learning relation extractor (Dong et al. 2005) and a hybrid relation extraction engine combining semi-supervised web based information (Yong and Su 2008) on ACE 2005. No feature engineering was done to adapt the systems for the dataset. The performance of the supervised extractor with devtest data (much larger than test data) on recall, precision and F-score are summarized in Table 1.1 under the columns marked as *Coref*.

On ACE 2004, the hybrid model improves performance up to 31% over the purely supervised one, but it does not improve performance much on the ACE 2005 data, because of the relatively large amount of training data. Such a large dataset might not be available in real applications however.

We evaluated the performance of the I2R relation extractor with and without coreference information. The supervised extractor uses features

	Recall		Precision		F	
	Coref	No	Coref	No	Coref	No
ART	0.517	0.25	0.744	0.7	0.61	0.37
GEN-AFF	0.586	0.576	0.783	0.854	0.67	0.69
ORG-AFF	0.753	0.675	0.791	0.866	0.77	0.76
PART-WHOLE	0.716	0.686	0.608	0.686	0.68	0.69
PER-SOC	0.758	0.192	0.833	0.76	0.79	0.31
PHYS	0.371	0.283	0.655	0.743	0.47	0.41

Table 1.1: Performance of I2R's supervised learning relation extraction engine on testing data with (Coref) and without (No) coreference chain information.

that require the semantic category and normalized headword information for entity mentions. The semantic category for pronouns and the normalized headword information for different mentions is derived from Named entities in the coreference chains. In other words, when we break all the coreference chains, we have no semantic tag information for pronouns and headwords are not normalized. With reference to the results shown in Table 1.1, removal of coreference information has a dramatic negative impact on F-score, up to 49%.

Relation extraction result has further been incorporated for CDC task on ACE corpus. The initial attempts have not shown much performance difference on CDC with or without relation information, mainly due to the easy nature of CDC ACE 2005, that is quite high performance is already acheived with simple features. Another reason is the data sparseness problem. There's not much repeat with the same types of relations from different news articles with ACE 2005, thus the information is too sparse to be useful for entity disambiguation. So different text collection might be easier to show the performance benifit from relation extraction on CDC, eg. a clean personal web page.

1.4 Intra-document coreference

Our work on intra-document coreference can be divided in three parts. First of all, we developed a platform for experimenting with intra-document coreference algorithms, the BART toolkit, whiche greatly facilitates testing different preprocessing models, different models of the coreference resolution process, and different types of features. Secondly, using this platform, we tested a variety of classifiers and models, above all examining the performance of Support Vector Machines with different types of kernels. Third, we developed and tested several methods for extracting lexical and encyclopedic knowledge from Wikipedia, the Web, and Wordnet, and different methods for deploying this knowledge. We briefly summarize our results in each of these areas here, referring to the specific chapters.

1.4.1 The BART toolkit

The Baltimore Anaphora Resolution Toolkit, or BART, is a highly modular and easily customizable platform for developing and testing fully automatic anaphora resolution models based on machine learning. Implemented in Java, it builds on the EMLR system developed by Ponzetto and Strube (2006) and incorporates ideas from the GUITAR system (Poesio and Kabadjov 2004) and from the work by Versley (2007) and Yang et al (Yang et al. To appear, 2006a).

BART makes it possible–in fact, relatively easy–to compare the results obtained using

- different preprocessing modules: during the workshop we tested both the YamCha chunker and Charniak and Johnson's reranking parser, and two Named Entity Recognition (NER) modules (the Stanford NER system and MITRE's Carafembic mention tagger;

- different models of IDC as a learning task: in addition to the standard model for generating training instances and for choosing an antecedent proposed by (Soon et al. 2001b) we tested a variety of alternative models, including ones using separate models for each type of NP;

- different classifiers, including C4.5 and SVMs;

- different sets of features: in addition to the set of features proposed by Soon et al, which we used as a baseline, we developed classes to extract features encoding lexical and encyclopedic knowledge, but also a variety of additional features, such as the tree features proposed by Yang et al. (2006a).

The system configuration to be used for a particular experiment can be modified in a declarative way

A crucial property of BART, inherited from the EMLR system, is that its input and output are encoded in the XML standoff format used by the

MMAX2 annotation tool (Müller and Strube 2003). This makes it very easy to compare the output of the system with the key (gold standard), facilitating error analysis.

Our experience with the toolkit was extremely positive; it will be made available in open source format via Sourceforge.

The architecture of BART and how to use it are discussed in some detail in Chapter 3.

1.4.2 Machine learning

One of the most important lessons of the workshop was that the additional information provided by the lexical and encyclopedic knowledge could not be fully exploited without addressing the limitations of the statistical model of IDC adopted by Soon *et al.*. Our efforts during the workshop focused on two areas: moving towards a classification scheme that would allow us to employ more complex features, such as SVMs; and testing alternative models of the anaphoric classification problem.

Kernel Methods for Coreference

The flexibility of kernel functions makes it possible to compute highly complex forms of similarity. During the workshop we took advantage of this opportunity to experiment new ways of modelling forms of similarity which have been claimed to play a role in intra-doc coreference, including:

- **binding constraints**. By representing the syntactic context in which the anaphor and potential antecedent occur, Yang et al. (2006a) were able to capture some of the restrictions on anaphoric reference known as 'binding constraints'–the fact that *him* cannot refer to *John* in *John likes him.*

- **syntactic parallelism**. Two mentions with the same syntactic position are more likely to corefer.

- **string** similarities for names. A variety of methods for computing similarities between names have been tried in IDC, such as minimum edit distance. Two possible ways of replacing this type of distance: **string kernels** and tree kernels between the 'parse trees' of the proper names.

Our research on using kernel methods for coreference is discussed in greater detail in Chapter 5

Models of the anaphora resolution process

In addition to the decoding scheme set forward by Soon et al., several others were implemented:

- The *split* decoder uses the basic Soon et al. scheme, but allows to use separate classifiers for pronouns and non-pronouns. This can help for SVMs, where training time grows superlinearly and testing time can grow linearly with the training set size, or to use classifiers with different properties for pronouns and non-pronouns.

- The *ranking* decoder uses a Maximum Entropy ranker to select candidates and has an adjustable resolution/non-resolution bias like the system described by (Luo et al. 2004). The use of a ranker-based model instead of a binary classifier has been found beneficial by (Versley 2006; Denis and Baldridge 2007). The ranking decoder is currently still a work in progress, as feature conjunctions (which exist for the MaxEnt classifier) have not been added to the ranker yet.

- The *stacked* decoder makes more linguistically motivated commitments, such as always treating indefinites as discourse-new, and combines a ranker to pre-sort candidates with a binary classifier that selects the best candidate(s) from the shortlist. The loss in recall that occurs through this pre-filtering is not made up by the precision gain, however, and results have been unsatisfactory so far.

1.4.3 Extracting Lexical and Commonsense Knowledge

Research on using lexical and encyclopedic knowledge for coreference has shown that any of the existing sources of commonsense knowledge or methods for extracting such knowledge by itself is incomplete and imprecise, but that better results can be obtained by combining knowledge extracted from several sources using this information as features for a supervised classifier, and letting the classifier choose which knowledge to use in the distinct cases. Thus, for instance, Poesio et al. (2004) used both knowledge from WordNet and knowledge automatically extracted from the Web using patterns; Ponzetto and Strube (2006) combined knowledge extracted from WordNet, Wikipedia, and using semantic role labelling; whereas Versley (2007) experimented combining GermaNet, information extracted using patterns using techniques similar to those proposed by Markert and Nissim (2005), and similarity measures.

We followed such approach as well, using features encoding knowledge extracted from WordNet, Wikipedia, and the Web. For WordNet we used the set of features developed by Ponzetto and Strube (2006), which will not be discussed here. Our work with web patterns was mostly concerned with adapting for English the techniques developed by Versley in earlier work, so will simply give a brief summary in this section. Most of our work was invested in extracting features from Wikipedia; this work will be discussed in some detail in chapter 4. All lexical and encyclopedic features are listed in the discussion of the BART system in Chapter 3.

Most work on using commonsense knowledge for coreference concentrates on improving recall for heads by identifying nouns that stand in relations of

- **instance** (e.g., *FCC / the agency*)
- **hyponymy** (e.g., *the retailer / the company*)
- **synonymy** (e.g., *the shop / the store*)

No work we are aware of attempts however to identify *in*compatibility between mentions: thus for instance we know that *FCC* and *AT&T* are certainly disjoint. We also experimented with using Wikipedia for this purpose.

Extraction from the Web

We used Web patterns to find **instance** relations such as those between *FCC* and *the agency*. Of the relations used in (Versley 2007), only the "*Xs* such as *Y*" and "*Y* and other *Xs*" are used, both for speed reasons and since the English WWW is usually large enough that just two patterns give enough coverage.

For every candidate / anaphor pair, where the anaphor is a definite noun phrase and the candidate is a proper name in the last 4 sentences, the extractor produces several pattern instances and calls Microsoft's Windows Live Search Web service to get a term count. By combining several patterns and using a Mutual Information threshold, it is possible to increase the precision of the extracted relations.

Features extracted from Wikipedia

In previous work, Ponzetto and Strube (2006) used the techniques discussed in Strube and Ponzetto (2006) to extract semantic similarity information from the Wikipedia category structure. We discovered that in the meantime information about categories in Wikipedia had grown so much and become so unwieldy as to limit its usefulness. Instead, we experimented with two novel techniques.

One set of methods extracted similarity features not directly from the category structure, but from a taxonomy constructed out of it and drastically filtered to remove, e.g., intermediate levels, using the methods discussed in (Ponzetto and Strube 2007).

A second set of methods extracted aliasing information from other sources of information, particularly hyperlinks and redirects.

Results suggest that both of these techniques perform about equally well (see Chapter 4).

Incompatibility models

The taxonomy extracted from Wikipedia by Ponzetto and Strube (2007) can also be used to compute information about incompatibility. In this taxonomy, distinct daughters of the same node –e.g., `India` and `United States`, both daughters of the category `Countries`–are typically incompatible. We use this structural information to extract incompatibility between mentions. In addition, we allow this information to percolate so that two mentions with incompatible modifiers–e.g., *software from India* and *software from the United States*–can also be found to be incompatible. Modifier incompatibility information is extracted from Wikipedia for mentions, and from WordNet for adjectives.

1.5 Evaluation and Annotation

The lack of adequate resources and of universally accepted evaluation metrics has always been one of the main problems for research in entity disambiguation. The availability of corpora for IDC has greatly improved in the last year, thanks in particular to the release of the OntoNotes corpus (Hovy et al. 2006), and substantial amounts of data for the Web People task have become available through the SEMEVAL competition and Spock challenge[3]. None of these resources however completely solves the problem, and no large annotated corpus was available for CDC. Therefore, we devoted a considerable amount of effort in this area. Specifically, we annotated the ACE 2005 corpus for cross-document coreference, and extended the ARRAU corpus with substantial amounts of text from the Penn Treebank in order to be able to

[3] `http://challenge.spock.com/`

test the systems developed in the workshop. Both resources will be made available through LDC.

As far as evaluation is concerned, there is no widespread agreement concerning the best evaluation measure either for IDC or for Web People. In addition to the original MUC score (Vilain et al. 1995) a number of metrics for entity disambiguation have become available, such as B-CUBED (Bagga and Baldwin 1998b) or the ACE scoring metric (Doddington 2001) but the community needs to agree on which measure is best.

We briefly summarize the work on annotation and scoring metrics here; a more detailed discussion is in Chapter 6.

1.5.1 The ACE CDC corpus

The ACE 2005 CDC corpus is an annotation of the ACE 2005 EDT corpus. The corpus is about 257K words and includes 55K mentions, which represent 18K distinct entities. The corpus was annotated using EDNA, an extension of the Callisto corpus annotation tool[4] implemented for the workshop.

1.5.2 The ARRAU IDC corpus

ARRAU[5] is a UK-funded project to explore hard cases in anaphoric interpretation, in particular, reference to abstract objects and ambiguous cases of reference. One of the objectives of the project is to create a medium-scale corpus annotated with a variety of intra-document anaphoric relations, including 'ambiguous' cases. Mentions are also annotated with a variety of additional information, and in particular, information about syntactic agreement. Most of the annotation prior to the workshop was of spoken dialogue data; for the workshop we added the annotation of around 40 Penn Treebank documents.

1.5.3 Scoring metrics for entity disambiguation

We carried out a theoretical analysis and comparison of several of the metrics proposed in the literature, and implemented a Java program that can compute them and is linked to BART.

[4] `http://callisto.mitre.org`

[5] `http://cswww.essex.ac.uk/Research/nle/arrau/`

1.6 Summary of Contributions and Conclusions

In summary, the main contributions of the workshop were as follows.

Concerning the Web People task, we demonstrated that non-greedy algorithms such as Metropolis-Hastings do outperform conventional greedy algorithms. We also found that lexical knowledge, in the format of topic models, results in further increases in performance.

Concerning IDC, we replicated results that tree kernels contribute to improvements in performance, and that automatically extracted lexical and encyclopedic knowledge result in such improvements as well. Furthermore, we found that the two results are cumulative, as these improvements affect different types of anaphoric expressions.

A more general goal of the workshop was to facilitate subsequent research in entity disambiguation by developing improved resources–both corpora and software. Quite a lot of effort was invested in this. We created the largest existing corpus annotated for cross-document coreference, and completed the annotation of the ARRAU IDC corpus, which includes texts of different genres, in which all types of nominal reference are annotated (including discourse deixis), and agreement information is annotated as well. On the software side we developed the EDNA annotation tool for CDC and the BART platform for experimenting with IDC.

Chapter 2

Web People

Rob Hall, Michael Wick, Jason Duncan and Paul McNamee

2.1 Introduction

A very common activity among Internet users is to issue a search where the query is a persons name. However, since names of people are often very ambiguous, it is usually the case that the returned search results will refer to many people (henceforth “web people”) sharing a common name. This is presumably a frustrating experience for internet users, since it places the burden on them to manually disambiguate the various pages. In this section, we seek to mitigate this frustration by learning functions which automatically perform disambiguation of web documents.

We describe a discriminative model which performs clustering, making use of first-order quantified features over clusters of documents, and learning parameters via a maximum entropy formulation. Inference in this model is performed approximately using a greedy algorithm, which is extended to a Metropolis-Hastings sampling scheme. Noting that labeled training data is hard to come by, we then develop an entirely unsupervised system which is generative and uses Gibbs sampling for inference.

Disambiguating web people is related to anaphora resolution insofar as it is inherently a clustering problem, however it is in many cases much harder. For example, the syntactic parse of a sentence often gives enough information to succeed at anaphora resolution, whereas a web document sometimes contains no obviously relevant information beyond a passing mention of the person’s name. In the SemEval 2007 web people task Artiles et al. (2007) it was demonstrated that named entities that appear in a locality around mentions to the web person can be very useful features for performing coreference Heyl and Neumann (2007); Popescu and Magnini (2007); del Valle-Agudo et al. (2007). In this work we extend this basic observation by employing more advanced clustering algorithms and more expressive feature space representations. We then explore the use of probabilistic topic models Blei et al. (2003) to gage the topical similarity between pages, in an attempt to gain further cues to coreference.

2.2 Problem Setting

We assume there are several sets of documents divided according to the ambiguous name which they contain. For example, all documents which refer to "John Smith" are in one set. In this work we make the assumption that each document refers to exactly one person. This is slightly different to the SemEval 2007 task Artiles et al. (2007) in which a document could refer to several people with the same name, however this restriction is reasonable considering that only very few documents (approximately 1% in the SemEval corpus) have this property (e.g., wikipedia disambiguation pages) which are not particularly interesting cases.

The problem of web-people coreference is to partition each set of documents into coreferent blocks. Therefore we seek to learn a predictor function:

$$f(x) = \arg\max_y P(x, y) = \arg\max_y P(y|x) \tag{2.1}$$

Where x is the set of documents, y is the coreference structure (i.e., a set of labels corresponding to x). We will describe a variety of formulations for $P(y|x)$ and $P(x, y)$, and corresponding ways to compute the maximization. In section 3 we will detail a discriminatively trained model, which requires hand-labeled data to perform parameter optimization. We drop the requirement of a labeled training corpus in section 4, when we detail an unsupervised generative model which performs the same task.

2.3 Discriminative Model

In this section we describe a recent model which combines first-order logic and probability, and that has been successfully applied to the intra-document coreference task. We adapt this model to disambiguating web people by aggregating features common to many of the successful SemEval Artiles et al.

(2007) systems under the formalism of first order logic. We explore the use of unsupervised topic models to obtain additional evidence that is not explicitly contained in the web documents themselves. Finally, we explore the deployment of statistically motivated clustering techniques to optimize these models. We demonstrate that topic models as well as more sophisticated clustering techniques yield improved results, indicating that both feature engineering and machine learning are two avenues to explore for performance gains in the web people tasks.

2.3.1 Probabilistic Model

The discriminative model is similar to the one used by Culotta et al. (2007) to tackle the newswire coreference task. In the newswire domain, this model factorizes into sets of mentions rather than just pairs of mentions, enabling more expressive features over larger sub-problems. As demonstrated below, it is straightforward to adapt this model to the case of web people.

Given a set of documents x^i, define a binary random variable y_i, such that $y_i = true$ if and only if all documents in x^i refer to the same underlying entity. We use a maximum entropy model for this binary classification decision: $p(y_i|x^i) = \frac{1}{Z_x} exp(\sum_k \lambda_k f(x^i, y_i))$ where λ_k are real-valued parameters, $f(x^i, y_i)$ are features over the set of documents x^i, and Z_x normalizes the distribution over the two labels. The values of $\Lambda = \lambda_1 \cdots \lambda_n$ can be learned from the labeled training data by performing gradient ascent.

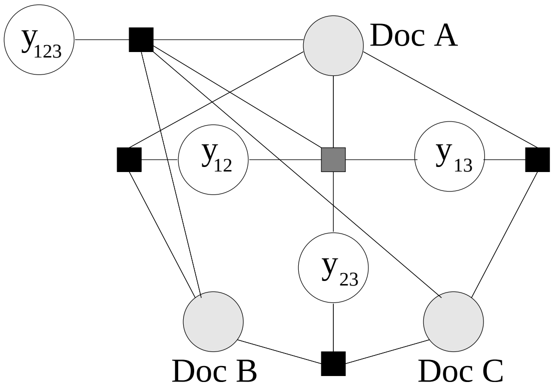


Figure 2.1: Factor graph representation of the coreference model with black boxes representing coreference (f_c) factors and gray boxes representing transitivity (f_t) factors

We can now define a probabilistic model over entire clusterings that factorizes into cluster-wise decisions as shown in Figure 2.1:

$$p(y|x) = \frac{1}{Z_x} \prod_{y_i \in y} f_c(y_i, x^i) \prod_{y_i \in y} f_t(y_i, x^j)$$

While the f_c factors represent the compatibility among clusters of documents, f_t factors ensure transitivity for all subsets of each cluster ($f_c = 1$ if transitivity is satisfied, $-\infty$ otherwise). Since there are a combinitorial number of f_t factors, this model cannot be fully instantiated. Additionally, the normalization constant Z requires summing over all possible cluster configurations, making approximate clustering techniques essential. The full model can be expressed as:

2.3.2 Features and Topic Models

Here we will discuss the feature functions used in our probabilistic model. We incorporate features used by many of the teams who competed in the 2007 SemEval tasks as well as those used by Mann and Yarowsky (2003). Such features include:
(1) cosine distance between bags of words
(2) term selection using TFIDF weights
(3) words in context windows around names in the document body
(4) cosine distance between chunks
(5) NER overlap
(6) n-gram matches in the web document title

A flaw in these features is that they require words between two documents to match exactly. For example, compare the following two excerpts about a fictional jazz musician, John Smith, from two different documents:

...his rhythmic punctuation...
...John's melodic improvisation...

Although there is no overlap between these excerpts, it is clear that both refer to John Smith, the jazz musician. Unfortunately, none of the aforementioned features are capable of providing evidence to allow the model draw this conclusion, since there is no word overlap. For this reason, we incorporate *topic* features that indicate whether two documents discuss common topics. More precisely, we use Latent Dirichlet Allocation (LDA) Blei et al. (2003), an entirely unsupervised topic model that infers mixtures of topics

Feature	InfoGain
TOPICS-top1-topic-not-same	0.24
ClusterSizeMoreThan8 & TOPICS-top1-topic-not-same	0.24
ClusterSizeMoreThan16 & TOPICS-top1-topic-not-same	0.23
ClusterSizeMoreThan16 & TFIDF-top10-no-matches	0.23
ClusterSizeMoreThan8 & TOPICS-top2-topics-not-same	0.22

Table 2.1: Four of the top five feature conjunctions contain topical evidence.

for each document. For each name set, we allow LDA to find 200 topics, and infer the corresponding mixtures of topics for each document. We then construct features that compare whether two documents have any of their highest weighted topics in common. The results are presented in Table 2.2, where we were able to show a 2% absolute increase in f1 by using topics as features. As seen in 2.1, the topic model features have some of the highest information gain.

Until this point, all the features described in this section involve the comparison of *t*wo web documents. However, we extend these pairwise comparisons into features over larger sets of documents by quanitifying and aggregating them with first order logic. For example, given a set of documents, and the feature function that checks if there is a 2-grams token match in the title (*2*-gram-title-match), the extension to first-order-logic features (over a larger set) include:

-There exists a pair of documents with *2*-gram-title-match

-There does not exists a pair of documents with *2*-gram-title-match

-For all documents: *2*-gram-title-match

-30% of documents have: *2*-gram-title-match

2.3.3 Clustering

A commonly used clustering algorithm is a hill climbing approach known as greedy agglomerative. The algorithm begins by placing each document into a singleton cluster. All pairs of clusters are compared and the two clusters with the highest compatibility score are merged. The compatibility scores between this newly formed cluster and the remaining clusters must be computed. The algorithm continues to greedily merge clusters until all the compatibility scores are below some threshold τ. With the maximum entropy classifier, $\tau = 0.5$ naturally falls at the decision boundary.

A major short-comming of this approach is that it can only modify the clustering by combining two complete clusters. If new evidence is discovered halfway through the clustering process that reveals an error, greedy agglomerative has no way of recovering. Additionally, the optimization surface is extremely bumpy, and greedy algorithms in general are likely to find maxima that are not global. This motivates the need for a less greedy clustering algorithm that has the ability to 'change it's mind'.

Metropolis-Hastings Metropolis et al. (1953); Hastings (1970) provides a framework for which arbitrary modifications can be made to the clustering. The algorithm works in rounds, making jumps in configuration space by drawing moves from a proposal distribution Q. The result is a sequence of clusterings $C_1 \cdots C_n$ corresponding to the n rounds of sampling. Let $C_t = y$ represent the configuration at time t, Metropolis-Hastings draws a new configuration y' conditioned on y from Q. The acceptance probability is then computed as $P(accept = true|y', y) = Min\left(\frac{P(y')Q(y|y')}{P(y)Q(y'|y)}, 1\right)$, and an acceptance decision is drawn from this distribution. Then $C_{t+1} = y'$ if accepted, y otherwise. Conveniently, the normalization constants cancel in this ratio, as well as the variables in the configuration space that remained unchanged between y and y'.

To avoid a slow burn-in time, we initialize Metropolis-Hastings with the result of greedy agglomerative clustering. Additionally, by keeping track of each agglomerative merge (and corresponding compatibility score) we can induce a probability distribution over partial clusters (or blocks), which is used as part of the proposal distribution in the Metropolis-Hastings phase. The proposal distribution Q is used to create a new clustering by (1) drawing a block from this block distribution, (2) removing the block from its original cluster, and (3) placing the block into another cluster. The move has four possible outcomes: either a new cluster is formed, part of one cluster is moved to another, a cluster is destroyed, or no change is made. Because at most, two clusters are modified in this operation, the acceptance ratio becomes a linear time computation. Let y'^s, y'^t be the source and target cluster in the proposed clustering and similarly y^s and y^t be the same clusters before the modification.

$$\frac{P(y')Q(y|y')}{P(y)P(y'|y)} = \frac{\frac{1}{Z_x}\sum_{i=0}^{n} exp\left(f(y'^i)^T\,\Phi\right) \sum_{j=i+1}^{n} exp\left(f(y'^i, y'^j)^T\,\Lambda\right)Q(y|y')}{\frac{1}{Z_x}\sum_{i=0}^{n} exp\left(f(y^i)^T\,\Phi\right) \sum_{j=i+1}^{n} exp\left(f(y^i, y'^j)^T\,\Lambda\right)Q(y'|y)}$$

	Precision	Recall	F1
BCubed	.32	.24	.28
+topics	.23	.44	.30
PW	.12	.19	.15
+topics	.13	.44	.20
MUC	.70	.65	.67
+topics	.84	.86	.85

Table 2.2: incorporating topics as features improves f1 in all three evaluation metrics

	Precision	Recall	F1
BCubed	.32	.31	.32
MH	.32	.44	.37
PW	.27	.24	.26
MH	.28	.37	.32
MUC	.84	.85	.84
MH	.86	.88	.87

Table 2.3: Metropolis-Hastings (MH) outperforms greedy agglomerative

$$= \frac{exp\,(f(y'^s)^T\,\Phi) + exp\,\left(f(y'^t)\right)^T\,\Phi) + \sum_{i=0}^{n} exp\,\left(f(y'^s y'^i)\right)^T\,\Lambda) + exp\,\left(f(y'^t y'^i)\right)^T\,\Lambda) + Q(y|y')}{exp\,(f(y^s)^T\,\Phi) + exp\,(f(y^t)^T\,\Phi) + \sum_{i=0}^{n} exp\,\left(f(y^s y^i)\right)^T\,\Lambda) + exp\,\left(f(y^t y^i)\right)^T\,\Lambda) + Q(y'|y)}$$

The partition function and the embedded sums cancel leaving only sums that require linear ($\Theta(n)$) time in the number of clusters.

2.3.4 Experimental Results

Training examples are created from the labeled dataset by randomly sampling documents with replacement. A set is labeled as a positive example if all documents in that set refer to the same underlying entity, and a negative example otherwise. We divided the 44 ambiguous name sets in the Spock corpus randomly into equal sized training and testing sets. For each of the 44 name sets, we sampled 150 random web documents. Coreference performance scores using MUC,B-Cubed, and Pairwise evaluation metrics are reported in table 2.3.

As we expected, the Metropolis-Hastings clusterer is able to improve substantially over its greedy counter-part. The clusterer is not only able to overcome local optima by making down-hill jumps, but it has potential to overcome these optima quickly by moving entire *blocks* during each round.

2.4 Generative Models

When experimenting with the discriminatively trained model, we noticed that the topical similarity features were weighted particularly highly. This implies that the results from the generative LDA model were quite indicative of coreference between documents. In this section we extend the basic LDA model to explicitly model coreference between documents via latent variables. The produced models are unsupervised, although training data could be used for optimization of hyperparameters, we chose to leave this for future research.

Since the number of clusters for a particular document set is unknown, use of traditional (parametric) mixture models is precluded, since they require setting the number of mixture components in advance. To allow for a variable number of clusters we use a mixture model with an infinite number of components. We employ a Dirichlet Process (DP) prior over the cluster assignments in each model. This is a distribution over clusterings parameterized by γ. A lower value for γ causes the prior to prefer singletons over fewer larger clusters.

2.4.1 Baseline Generative Model

The simplest generative model we considered does not model documents as mixtures of topics as in LDA. Rather it assumes each document belongs to some cluster (the identity of which is latent) and each cluster has one topic associated with it. This graphical model is shown in figure 2.2. The model defines a distribution of the form:

$$P(w, e|\alpha, \gamma) = \prod_d P(w_d|\theta, e_d) \cdot \prod_i P(\theta_i|\alpha) \cdot P(e|\gamma) \tag{2.2}$$

Where i is the index of the cluster, and d is the index of the documents, w_d are the tokens of document d, e is the set of clusters, and θ_i are the parameters to the mixture component associated with cluster $e_d = i$. $P(w|\theta, e)$ is the probability of the document tokens given its cluster identity and the associated parameter vector θ_i, and is a multinomial distribution. $P(\theta_i|\alpha)$ is the probability of the multinomial parameter vector which is expressed as

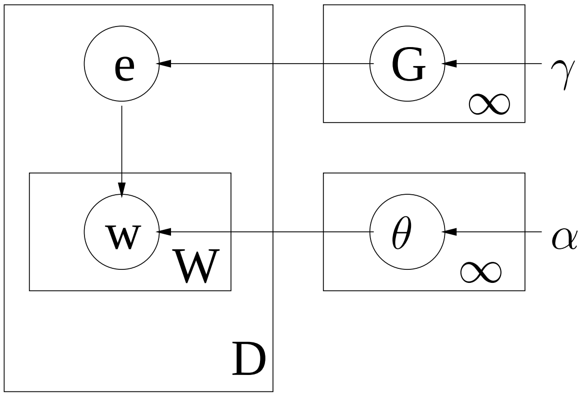


Figure 2.2: The baseline generative model: each of D documents has an entity e drawn form a DP prior, with parameter γ. The W words in the document are drawn from the multinomial distribution θ according to its entity setting.

a Dirichlet prior. $P(e|\gamma)$ is the DP prior over the cluster assignments. By changing the value of γ the prior can be made to favor different granularities of clusterings.

To perform inference in this model we use collapsed Gibbs sampling. It is collapsed in the sense that we integrate out the unknown θ variables in order to reduce the space in which we have to sample. Since the Dirichlet is the conjugate prior to the multinomial distribution, we are able to do this integration analytically. In doing so, all the words under a common entity become dependent on each other. The result is a compound distribution sometimes called the Polya distribution Minka (2003). The probability of the words in document d; $w_d = w_{d,1} \cdots w_{d,n_d}$ given a cluster is then:

$$P(w_d|w_{-d}, e) \propto \prod_i \frac{\alpha_{w_{d,i}} + \sum_{p|e_p=e_d} n_{w_p,w_{d,i}} + n_{w_{d,1\cdots i-1},w_{d,i}}}{\sum_v \alpha_v + \sum_{p|e_p=e_d} n_{w_p,v} + n_{w_{d,1\cdots i-1},v}} \tag{2.3}$$

Here $n_{w_p,w_{d,i}}$ is the number of times that the word $w_{d,i}$ appears in w_p (a set of words representing a document in the same cluster). Likewise $n_{w_{d,1\cdots i-1},v}$ is the number of times word v appears in the set of words $w_{d,1} \cdots w_{d,i-1}$. Both can easily be calculated by maintaining a set of the necessary counts during sampling.

Gibbs sampling, updates one hidden variable at a time, by sampling from the its distribution, conditioned on the current assignments to all other variables. During gibbs sampling, we use the Chinese Restaurant Process Neal (1998) construction for sampling the DP prior. The CRP gives mass to each cluster, proportional to the number of elements in that cluster. Each e variable can be set to either one of the currently supported clusters, or to a "new" cluster, which has the mass for the infinitely many unsupported mixture components in the model. When there are N documents divided into K clusters, the CRP gives the probability of document d being in cluster i as:

$$P(e_d = i | e_{-i}) = \begin{cases} \frac{n_i}{N-1+\gamma} & i \in 1 \cdots K \\ \frac{\gamma}{N-1+\gamma} & i = K+1 \end{cases} \tag{2.4}$$

We will use e_{-i} to mean the set $\{e_d | d \neq i\}$. Gibbs sampling iterates over each document d in turn and re-samples its value of e_d, from the distribution:

$$P(e_d = i, w_d | w_{-d}, e_{-d}) = P(w_d | w_{-d}, e) \cdot P(e_d = i | e_{-i}) \tag{2.5}$$

This model essentially clusters documents based on the distributions of words within them. For example if two documents employ many of the same words, they are likely to be placed into a cluster together.

2.4.2 Extended Generative Models

The baseline model only considers the tokens that appear in documents. Consequently it has two significant weaknesses. First, there are more sources of information regarding coreference than just the tokens of a document. For example, the hypertext markup such as links and bolded sections might give important clues to coreference. Second, various words in a document might not be relevant to coreference resolution. Common English words such as "the", "a", "and" etc. are presumably poor predictors of coreference. However since the baseline only looks at the distributions of tokens within a document, those documents with similar distributions of these "stop-words" might be placed together erroneously. Motivated by apparent weaknesses of the baseline model, we now explore a variety of extensions that will be able to capture more of the cues to coreference that the discriminative model employs.

LDA Coreference Model

The first extension grants the model the ability to learn that certain words are more relevant to coreference resolution than others. Rather than each

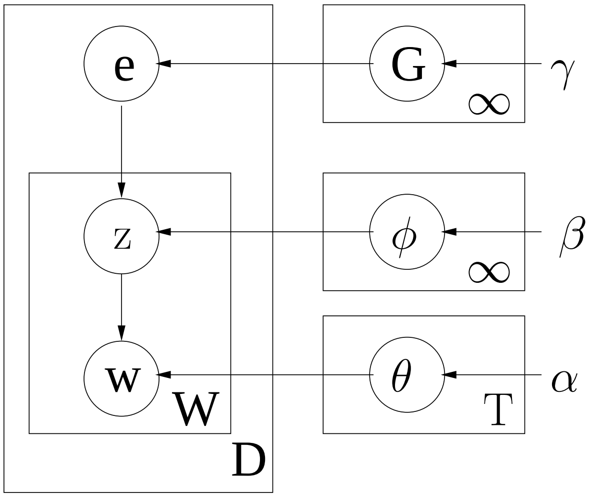


Figure 2.3: The “LDA-coref” generative model: now the words in the document have associated “topic assignments” z. Clustering is performed based on the topics that appear in the documents, rather than the tokens themselves.

entity having a distribution over document tokens, we assume each document is a mixture of words from some LDA topics, and each entity has a distribution over topics. The intent is to learn clusters of words that are related, and treat all words in the topic as the same with regard to coreference. The graphical model is shown in figure 2.3.

To perform inference we must alternate between sampling the topic assignments to the words, and sampling the entity assignments of the documents. The former are sampled from the following distribution:

$$P(z_{d,i} = t | z_{-d,i}, w, e) \propto \beta_t + \sum_{e_p = e_d} n_{e_d,t} \frac{\alpha_{w_{d,i}} + n_{t,w_{d,i}}}{\sum_{w'} \alpha_{w'} + n_{t,w'}} \tag{2.6}$$

With all the topic assignments z sampled, we can condition on them while sampling new entity assignments e from:

$$P(e_d = i, w_d | w_{-d}, e_{-d}, z) \propto P(z_d | z_{-d}, e) \cdot P(e_d = i | e_{-i}) \tag{2.7}$$

Note the similarity between equations 2.5 and 2.7. We are essentially treating the topic variables in this model as we did the words in the baseline. This model will therefore cluster documents according to the topic

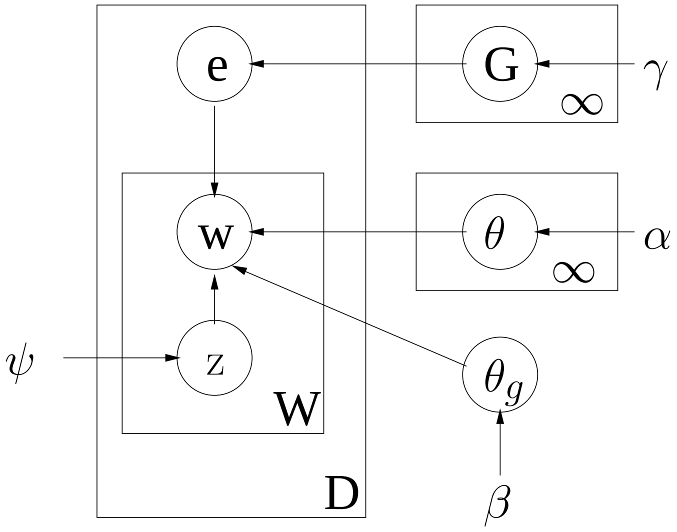


Figure 2.4: The "self-stopping" model: words either belong to an entity specific "topic" (as in the baseline) or to the global topic θ_g. Clustering is performed based on the words that belong to the entity specific topics.

distributions inferred on them. The intuition for this model is that words that indicate coreference might not overlap between documents (for example, one document mentions jazz music while another mentions saxophone playing), but by clustering words into LDA-like topics these words will all be viewed as the same by the model.

Self-Stopping Model

The LDA-like model of the preceding section seems appealing for its ability to distinguish between words of different indicative power for coreference. However with large numbers of topics it becomes computationally expensive to perform sampling. Therefore we propose a simplified version that aims to capture this distinction in a more efficient fashion.

We propose to use a topic-like mechanism to capture the difference between background English language "noise" and the entity specific language that is useful in coreference. Therefore we use a restricted form of the previous model, where we set the number of topics to be the number of entities plus one (note that this number changes as clusters are created or destroyed in sampling). There is one entity-specific topic for each entity, and one "global" topic that is shared between all entities, and is depicted in figure 2.4. Now each document is treated as a mixture of words from the entity-specific topic for the cluster to which it is assigned, and the global topic. This means that sampling the z variables is simpler, since there are now binary. The effect of the model is to cluster documents which share similar distributions of words from the entity specific topic. This model is unable to capture some of the properties exploited by the LDA-coreference model, but may still be able to filter out the noise in a document. We call it the self-stopping model for its ability to ignore stop-words when making the coreference decisions.

Incorporating Further Evidence

In designing the discriminatively trained model, we noted that HTML markup elements such as hyperlinks and page titles were important features for determining coreference. We may extend any of the generative models we described so far to use this type of evidence as well. If we assume are other forms of evidence are expressed as tokens in the documents (e.g., hyperlink URLs, hyperlink text, document titles) then we can extend the models to observe these as well. In essence, each of the models we showed can be extended to handle this evidence, by duplicating the components which involve words (and their associated topic variables and parameters) for each other evidence class. Note that for several classes of evidence, the values of the parameters used for words may not work well. For instance coreferent documents may have no hyperlinks in common, and so a higher (more uniform) value for the Dirichlet prior parameter may be more appropriate.

2.4.3 Results

We evaluated these models on the new Spock corpus described in chapter 6. We present a preliminary result clustering the group of documents belonging to the “Peggy Waterfall” web-people. This section has 1302 documents, divided into 91 unique people. For all models, we only used the tokens from the documents which were within 50 words of a mention to the web-person name. This was due to the intuition that relevant information appears close to the name mentions. All models had the γ parameter set to 0.005. We tested the systems under the following configurations:

- **Baseline** The generative model described in section 2.4.1. This model was given an α parameter of 0.01 since we anticipated that the distributions of words for the entities should be peaked around certain indicative words.

- **LDA-coref** The LDA-inspired model described in section 2.4.2. We used 200 “topics”, and set α to 0.25 and β to 0.01. Note that the β parameter here corresponds to the α parameter of the other models.

- **Self-stopping** This is the model from section 2.4.2. The parameter for the “global topic” was set to the unigram counts of the corresponding words in the corpus. The intent was to capture the highly frequent words in this topic. α and γ were set as in the baseline model.

- **Baseline+URL** As described in section 2.4.2, we extended the baseline model to also observe the bags of hyperlink URLs for each document, as well as the bags of words that make up the link text. Now the model has an α parameter for each “class” of evidence, words, URLs and link text. We set these to 0.01, 0.9 and 0.9 respectively.

- **Self-stopping+URL** This is a hybrid of the above two systems. Essentially we extended the self-stopping model to observe the other two evidence classes that we described above. The setting of α was the same as for Baseline+URL, and for each class, we learned a global topic parameter by taking the unigram counts of the evidence tokens.

All models were trained with 200 iterations of Gibbs sampling. When there were both topic variables and entity variables, topic variables were sampled first for the whole corpus, before entities were resampled. Results of the various systems are shown in table **??**. They show a trend whereby the incremental improvements yield small increases over the baseline model. The LDA model lags behind considerably, perhaps due to a bad parameter setting, or to an insufficient amount of Gibbs sampling. However note that the documents often contain a mention to the web-person surrounded by useless text (such as when the persons name appears in a table of sports results). These cases might prevent the proper learning of a topic model such as LDA. These results are preliminary in the sense that they only cover one cluster of the Spock data, however they should be updated in the near future. The results are generally quite low, but note that they are not a great deal lower than those of the discriminatively trained model, and that this is evaluated on the entirety of one of the largest sections of the spock data – rather than a sample of 150 documents from each section.

	B^3			Pairwise		
Model	Precision	Recall	F1	Precision	Recall	F1
Baseline	63.3	16.4	26.0	39.2	9.8	15.7
LDA-coref	37.8	7.83	13.0	48.1	14.1	21.8
Self-stopping	69.7	17.3	27.7	60.8	9.6	16.5
Baseline+URL	51.7	18.2	26.9	12.1	11.9	12.0
Self-stopping+URL	60.9	18.6	28.5	53.2	11.1	18.4

Table 2.4: Coreference performance for a selection of generative web-people disambiguation models.

2.5 Future Work

We have demonstrated a variety of techniques for disambiguating web documents according to the people to whom they refer. There are several directions in which these models could be extended, for example exploring more powerful features, more sophisticated statistical models, and more advanced inference and sampling algorithms. Additionally we may gain benefit from combining the unsupervised method into the discriminatively trained one. We noticed that the two models have different strengths and weaknesses, for example the unsupervised model performs well on large corpora, whereas the discriminative model is more accurate when there are fewer web documents to cluster. A successful line of future research may combine the models by using the predictions of the unsupervised model as features for the discriminative one.

This problem lends itself to semi-supervised learning, in which a small amount of labeled data is combined with a vast amount of unlabeled data in training a predictor. Due to the sheer sizee of the Internet, such unsupervised data would be readily available in an almost unlimited supply.

In the related problem of anaphora resolution we have successfully applied error driven training methods Culotta et al. (2007), which tailor the parameters to avoid the types of errors that the model produces during training. Such techniques are directly applicable to this problem, and may increase performance over the current models. Furthermore, the discriminative algorithm we outlined factorized over clusters of documents. A straightforward extension would be to relax this restriction and allow for features that consider an entire clustering.

Chapter 3

Description of the Elkfed/IDC platform and the BART coreference resolver

Yannick Versley and Simone Ponzetto

3.1 General Introduction

BART, the Baltimore Anaphora Resolution Toolkit, is a tool to perform fully automatic machine-learning based automatic coreference annotation on written text. This section will provide a friendly introduction to the system from a user's perspective.

The system stores all vital information on documents in the token-based standoff format of MMAX2; it uses the MMAX2 discourse API[1] for this purpose.

In the standard configuration, only tokenisation is needed, and other steps are performed automatically by suitable components (sentence splitter, part-of-speech tagger, chunker/parser, and named entity recognizer). For learning a new classifier or quantitative evaluation, it is necessary to have gold standard coreference information on a separate markable level.

To run the basic system, you need to have the following external components installed:

- the YamCha chunker and the YamCha model collection (for the chunker-based pipeline)

[1] see http://mmax2.sourceforge.net

`http://chasen.org/~taku/software/yamcha/`
(where do the chunking models come from?)

YamCha uses an external SVM package to perform its classification; possible candidates are TinySVM and SVMLight
`http://chasen.org/~taku/software/TinySVM/`
SVMLight/TK, which is a downwards-compatible extension to SVMlight, can also be used as a learner in the coreference resolution.

- Charniak and Johnson's reranking parser
ftp://ftp.cs.brown.edu/pub/nlparser/

Other recommended external components include

- SVMlight/TK with java native interface: This allows the use of SVM for classification tasks and the use of tree-valued features. The use of the native Java interface is recommended for improved speed.
- The Carafembic ACE mention tagger performs general mention tagging for ACE mentions. Its use improves the accuracy when using ACE-style corpora in which only ACE mentions (persons, organizations, geopolitical entities, ...) are marked up.

The *preprocessing pipeline* invokes sentence splitter, part-of-speech tagger, chunker and named-entity recognizer and uses this information to tag mention markables (on the *markable* annotation layer of the MMAX2 document). Once documents have been preprocessed, the preprocessing information in the MMAX documents can simply be reused and preprocessing switched off. This is especially convenient when doing repeated experiments on a single dataset.

3.1.1 Installation and Getting Started

This section will lead us through the steps necessary for running training and testing phases on the MUC6 corpus.

Running without preprocessing

1. In the directory `config`, make a copy of the file `config.properties.sample` and name it `config.properties`.

   The `config.properties` file contains configuration options that usually depend on the local system configuration, such as the directories where training/testing data, needed programs, etc. reside.

If you unpacked the MUC sample files into `/path/to/MUC-MMAX`, then you need to set the trainData/testData options as follows:

```
trainData = /path/to/MUC-MMAX/muc6/train
trainDataId = MUC6
testData = /path/to/MUC-MMAX/muc6/test
testDataId = MUC6
```

In the run we want to do now, we don't need to run the preprocessing, as the MUC files are already in MMAX format and conveniently preprocessed:

```
runPipeline=false
```

MUC6 marks coreference even outside the main document body, which is why we want to use the mention creation process that uses mentions from the whole document:

```
mentionFactory=elkfed.coref.mentions.FullDocMentionFactory
```

2. To compile the Elkfed sources, we need to have (i) a working JDK (version 5.0 or up) and (ii) Apache Ant[2]; we also need to setup the classpath so that external libraries (which are part of the Elkfed/BART package) can be found.

   First, edit the `setup.sh` file so that `JAVA_HOME` points to the directory where your Java installation is. You then need to source the file with[3]:

   ```
   bash$ source setup.sh
   ```

   We then run ant to compile the whole thing:

   ```
   bash$ ant jar
   ```

   We can then use `XMLExperiment` to perform both training and testing[4]:

   ```
   bash$ java -Xmx1024M elkfed.main.XMLExperiment
   ```

[2] available at `http://ant.apache.org/bindownload.cgi`

[3] this only works with bash. Users of other shells such as tcsh will have to adapt this.

[4] the option -Xmx1024M is used to allocate more heap space for the Java process. If your computer does not have enough memory, or Java runs into memory problems, you have to adjust this number

or we can use XMLTrainer to create the training data, run XMLClassifierBuilder to perform model learning and then use XMLAnnotator to test separately:

```
bash$ java -Xmx1024M elkfed.main.XMLTrainer
(lots of output omitted)
bash$ java -Xmx1024M elkfed.main.XMLClassifierBuilder
(some output omitted)
bash$ java -Xmx1024M elkfed.main.XMLAnnotator
(lots of output omitted)
```

Running with the parser pipeline

To try out some preprocessing, we will first use the ACE-02 sample file that is in sample/ACE-02[5]. To do this, we first change the testData configuration entry in config.properties:

```
testData=./sample/ACE-02
```

We then need to change the options so that (i) preprocessing is activated, (ii) the Charniak parser is used and (iii) the directory where the Charniak parser is located is known to the system:

```
runPipeline=true
pipeline=elkfed.mmax.pipeline.ParserPipeline
parser=elkfed.mmax.pipeline.CharniakParser
charniakDir=/path/to/the/reranking-parser
```

To be able to use the Charniak parser, we also need to replace the parse.sh script in the reranking-parser directory with our modified version.

We can then run XMLAnnotator, which uses the model we trained on the MUC data (*again, this is not useful for any serious purpose, but we want to try out the pipeline*) on the corpus that will be run through the preprocessing pipeline for us.

3.1.2 Additional Configuration

The *config.properties* file in the *config* directory contains a few more settings that influence the behaviour of the system:

[5] This is nonsense from an evaluation point of view, as the ACE and MUC annotation schemes differ considerably. But as preprocessing the whole MUC6 corpus would take longer, we'll just have fun with the sample file.

- the option `mentionFactory` indicates the name of the class used for creating the mention objects from MMAX markables, which can be used to influence the set of mentions that are created and can then be linked.

  Currently, the following *MentionFactory* subclasses exist[6]:

  - `FullDocMentionFactory` creates mentions for every markable on the *markable* annotation layer
  - `DefaultMentionFactory` creates mentions for every markable that is in the 'main text' part (marked by a markable on the *section* annotation layer with attribute *name*=*text*).

- The value of *trainDataId* / *testDataId* selects the following corpus-specific behaviour:

  - If either trainDataId or testDataId are set to `MUC6`, the anaphor must be a definite for the expression to be an apposition (in `FE_Appositive`).

- The value of *runPipeline* can be set to *true* if it is desired to (re-)run the preprocessing steps on the corpus, or *false*, if existing annotation layers are to be reused.

- The value of *pipeline* can be used to select a different version of the preprocessing pipeline[7]:

  - `DefaultPipeline` uses a the Stanford POS tagger, the YamCha chunker and the Stanford named entity tagger.
  - `ParserPipeline` uses the Charniak parser to extract POS tags, BaseNPs as chunks, and also extracts parse trees.
  - `NERTestPipeline` uses the Charniak parser to extract syntactic structure, but uses the Carafembic mention tagger for extracting both nominal and name mentions. Because only ACE entities are extracted and non-ACE noun phrases are ignored, this is the recommended preprocessing when using ACE-style corpora which do not mark all mentions.

[6] it is necessary to prepend the package name *elkfed.coref.mentions* in all cases

[7] it is always necessary to prepend the package name *elkfed.mmax.pipeline*

- The value of *default_ system* determines the feature set and learners to be used. To use different settings, it is possible to either give XMLExperiment the name of an XML file containing such a description, or change the value of *default_ system* to the name (without the .xml suffix) of an existing description from the *elkfed.main* package. In the current distribution of Elkfed/BART, the following XML descriptions are included:
  - `idc0_system` uses exactly the Soon et al. feature set (mention type, gender/number agreement, alias, appositive, semantic class compatibility, sentence distance).
  - `bart_system` uses an extended feature set: besides the information used by IDC0, it also uses parse tree information (tree kernels, syntactic position), as well as some semantic information (web patterns, Wikipedia alias, semantic class values).

    The BART system uses tree kernels and requires external information (web queries and information extracted from Wikipedia in a relational database), which means that setting it up requires some work. For more details, please refer to the descriptions of the individual features in section 3.3.

3.1.3 XML system descriptions

The encoding/decoding model used as well as the learners and the features used can be influenced by means of XML description files. The two description files that can be used out of the box are loaded from the JAR file; they can be found in the package `elkfed.main`, whereas other examples can be found in the package `elkfed.main.old_xml`. To use an alternative system description, just put it in the current directory and give the filename to Xml{Trainer/Annotator/Experiment}.

Figure 3.1 shows the system description for the IDC0 system. The root element, `coref-experiment`, has exactly one `system` node, which in turn has a list of classifiers and a list of extractors. In the `soon` system type, the only we will cover here, we only need one classifier, which is used for all anaphor-antecedent pairs.

The following classifieres are implemented:

- The `weka` classifier uses the WEKA machine learning toolkit for classification; all classifiers from WEKA can be used, and the class name of the corresponding classifier has to be given in the “learner” attribute.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<coref-experiment>
<system type="soon">
    <classifiers>
        <classifier type="weka" model="idc0"
            learner="weka.classifiers.trees.J48"
            options=""/>
    </classifiers>
    <extractors>
        <!-- general info about antecedent -->
        <extractor name="FE_MentionType_Buggy"/>
        <!-- agreement features -->
        <extractor name="FE_Gender"/>
        <extractor name="FE_Number"/>
        <!-- specialized features for aliases etc. -->
        <extractor name="FE_Alias"/>
        <extractor name="FE_Appositive"/>
        <!-- string matching features -->
        <extractor name="FE_StringMatch"/>
        <!-- semantic class agreement -->
        <extractor name="FE_SemanticClass"/>
        <extractor name="FE_SentenceDistance"/>
    </extractors>
</system>
</coref-experiment>
```

Figure 3.1: XML system description: IDC0

Options, as they appear on the command line shown by the WEKA Experimenter, can be specified in the "options" attribute.

- The `svmlight` classifier uses SVMLight, either in its plain variant or in the SVMLight/TK variant. Options to `svm_learn` can be specified in the "options" attribute.
- The `maxent` classifier is a maximum entropy classifier built upon the L-BFGS implementation of Mallet. It is able to perform feature combinations. Binary feature combinations give you a similar accuracy to the SVMLight polynomial-degree-2 classifier, with much reduced training times.

> The preliminary interface for this is that the “options” attribute is interpreted as a combination template, i.e. `options="**"` uses the features alone, whereas `options="** **"` gives binary feature combinations. **This is subject to change. Use with care!**

The extractors are listed in section 3.3; the name of a feature extractor is specified in the “name” attribute and a matching class is then searched for in the package *elkfed.coref.features.pairs* and in the subpackages *elkfed.coref.features.pairs.{srl/wiki/wn}*

3.2 Inside BART: architecture and internal APIs

One goal for the Elkfed architecture has been to provide effective separation of concerns for the following three groups of people who might be interested in working on a system for coreference resolution:

- Those who aim to do *feature* engineering, creating new features that exploit different sources of knowledge.
- Those who aim to explore different *preprocessing* methods, improving the quality of the input to coreference resolution proper.
- Those who aim to explore different methods of representing coreference resolution as a *learning* problem.

To reach this goal, there is a clean separation between the domains of preprocessing, feature extraction, and learning:

The first part of preprocessing is carried out by pipeline components, which add MMAX markables on different annotation layers, and stores the result on the *markable* annotation layer in MMAX. The second part of preprocessing, carried out by `MentionFactory` instances, uses the markables on the *markable* annotation layer to create Java objects with relevant properties, instances of class `Mention`.

Feature extractors are presented are presented instances of the relevant `Instance` subclass — in BART, which exclusively uses binary decisions, this is always `PairInstance`. They then use the information stored in the `Instance`, namely the *anaphor* and *antecedent* properties, which hold references to mention objects. Having each feature extractor in its own class allows for flexible mixing and matching for feature extractors.

The part that is responsible for learning decision functions using a given set of features (referred to as the encoder/decoder) uses a machine learning classifier from the *elkfed.ml* package that is trained with anaphor - potential antecedent pairs from the training set, and the decisions of this classifier regarding single pairs are then used to derive appropriate linking decisions that group mentions into equivalence sets representing entities. The encoder/decoder has to extract pairs that are to be presented to the learner, and delegate the feature extraction to a list of feature extractors. In the testing phase, it has to choose pairs to present to the classifier built in the training phase and to use the classifier decisions to link mentions.

3.2.1 Important Classes

The most basic building blocks in the Elkfed platform are the interfaces `CorefResolver` and `CorefTrainer` in the package *elkfed.coref*. A coreference resolver get handed a list of `Mention` objects that are to be grouped together in a `DisjointSet`, whereas a `CorefTrainer` just gets handed the list of mentions and is not required to return anything.

`Mention` objects represent single mentions: they have utility methods that allow to access properties of mentions, and a method `isCoreferent` that allows the training procedure in a ML-based coreference resolution system to see whether a pair of mentions should be coreferent or not.

What happens around these interfaces? Let us begin by the outer side: in the package *elkfed.main*, the classes `Trainer` and `Annotator` are simplified versions of BART's XMLTrainer and XMLAnnotator classes and contain the necessary code for setting up the actual process.

Objects of type `SoonEncoder` or `SoonDecoder` (to be covered later, below) are handed to instances of `TrainerProcessor`, or `AnnotationProcessor`, respectively, that iterate through documents in the corpus given and then use a *MentionFactory* to create `Mention` objects from the information in the MMAX2 documents.

`SoonEncoder` instances take a list of markables; for every pair m_j, m_i of mentions that are adjacent in a coreference chain, a positive training instance is generated for the pair $\langle m_j, m_i \rangle$, and a negative instance with $\langle m_k, m_i \rangle$ is created for every markable m_k that occurs in between m_i and m_j. These learning instances serve as learning data set for the ML classifier; an object implementing the *IntanceWriter* interface takes these instances and writes them out in a format that is understood by the ML toolkit implementing that classifier, for example in ARFF format for Weka-based learners.

In converse, `SoonDecoder` instances look for an antecedent for a given markable m_i by getting the classification for pairs $\langle m_j, m_i \rangle$ built with some m_j that occurs before m_i, starting with the closest ones; the first pair to be classified as positive is merged and other (potential) antecedents are ignored. The classification of pairs is handled by an object implementing the *OfflineClassifier* interface, which gets a list of pairs and provides the list of decisions for these pairs. In the case of the Weka machine learning toolkit, the classifier is called in-process. For classifiers that are only available as external programs (such as SVMlight when the native interface is not used), always classifying batches of multiple pairs attenuates the speed loss due to the startup time of the external program.

The classification instances that are used for learning and classification are instances of the class `PairInstance`, which get the anaphor and antecedent set by the encoder/decoder, whereas the actual information used for classification is set by objects implementing the `PairFeatureExtractor` interface.

3.3 Feature Extractors

This section describes the feature extractors that are included in the Elkfed/IDC platform; most, but not all of them are used by BART. Tree-valued features can only be used by the SVMlight learner, string-valued features cannot be used with WEKA learners, and unnormalized continuous features do not work well with polynomial SVMs or MaxEnt classifiers that use feature combinations, so not all sets of features make sense with a given learner.

3.3.1 Basic Features

MentionType

The feature extractors `FE_MentionType_Buggy` and `FE_MentionType` extract information about the form of the anaphor (definiteness, demonstrative, pronoun), the antecedent (pronoun) and also includes a feature that indicates whether the two mentions are both proper names.

`FE_MentionType_Buggy` checks for the prefix “the” on the mention string to derive definiteness, whereas the *isDefinite* method on Mention checks that “the” is actually a word by itself, excluding “them”, “their” and other third person plural pronouns. In the basic Soon et al reimplementation, the information found in the ‘buggy’ version (third-person plural pronouns) is used and leads to improved performance over the corrected version.

Gender agreement

The feature extractor `FE_Gender` uses gender information from the mention to assess gender compatibility. The assigned value can either be true, false, or unknown.

Number agreement

The feature extractor `FE_Number` uses number information to determine number compatibility. This is either true or false.

Alias

`FE_Alias` uses the techniques described in (Soon et al. 2001b) to match abbreviations and name variations.

Appositive

`FE_Appositive` adds a feature that is true whenever two mentions are separated exactly by a comma.

String Matching

`FE_StringMatch` strips the determiners off the markable string and then performs a case-insensitive comparison of the rest.

Semantic Class compatibility

`FE_SemanticClass` uses the SemanticClass property of the mention to assess the semantic compatibility of anaphor and antecedent (either TRUE, FALSE, or UNKNOWN if either of the two has an unknown semantic class and the lexical heads do not match).

Sentence distance (continuous vs. discrete version)

`FE_SentenceDistance` gives the distance of anaphor and antecedent candidate in sentences. `FE_DistDiscrete` is meant as a discretisation of the values, with two binary features that indicate whether the candidate is in the same sentence or in the previous sentence.

3.3.2 Syntax-based Features

Syntactic position

FE_SynPos yields a string that is composed of the first three unique labels of parent nodes. This is meant to indicate the syntactic position — subjects will have a value of 'np.s', whereas direct objects will have a value of 'np.vp.s', and a noun phrase embedded in a noun-modifying PP would have a value of 'np.pp.np'.

Tree features

The feature FE_TreeFeature is a tree-valued feature that carries information about the syntactic relationship between anaphor and candidate. Its value is a subtree of a parse tree covering both the anaphor and the antecedent candidate. It includes the nodes occurring in the shortest path connecting the pronoun and the candidate, via the nearest commonly dominating node. Also it includes the first-level children of the nodes in the path.

3.3.3 Knowledge-based Features

Web patterns

The FE_WebPatterns feature extractor uses pattern search on the World Wide Web to find instance relations as they exist between 'China' and 'country', or 'Clinton' and 'president'. Queries are cached in a local BerkeleyDB-JE database.

The following settings in config.properties are necessary for this feature extractor to work:

- **msn_app_id** contains the developer key for Microsoft's Windows Live Search service. The process of getting a developer key for this service is described at the following URL:
  http://dev.live.com/blogs/livesearch/archive/2006/03/23/27.aspx

Results of web queries are cached in a Berkeley DB Java Edition database, which is created in the current directory. The current implementation unfortunately precludes concurrent access from multiple processes on the same file system, but a cache that has been established once (by doing the queries needed) can simply be moved to another machine by copying the *.jdb files.

Wikipedia Alias

The `FE_WikiAlias` feature extractor uses information extracted from Wikipedia[8], namely redirects and links to a given page, but also appearance in lists, to provide evidence for name variations (see the extraction chapter for a more detailed description).

The Wikipedia Alias feature extractor needs to access a MySQL[9] database that contains the *redirects_ to*, *links_ to* and *lists_ dev* tables with information from Wikipedia. The following settings in config.properties are necessary for this feature extractor to work:

- **wikiDB _driver** contains the class name of the JDBC driver, usually *com.mysql. jdbc.Driver*
- **wikiDB _user** and **wikiDB _password** contain user name and password of the account that is used to connect to the database
- **wikiDB _dburl** contains the JDBC URL to the database. This should be somethink like
  ```
  jdbc:mysql://⟨hostname⟩:3306/⟨database⟩↩
  ?useOldUTF8Behavior=true&useUnicode=true&↩
  characterEncoding=UTF-8
  ```
  (without the line breaks or ↩ in between the parts).

Wiki (category graph)

The `FE_Wiki` feature extractor uses redirects and the category graph of Wikipedia to assess candidate relatedness, as described in (Ponzetto and Strube 2006). See the chapter on knowledge extraction for a more detailed description.

Wordnet distance

The `FE_WNSimilarity` feature extractor extracts the WordNet distance between antecedent and candidate heads, according to several distance measures.

SemClass pair

`FE_SemClassValue` extracts the semantic class values of anaphor and antecedent, both alone and as a pair.

[8] see *http://www.wikipedia.org*

[9] it is probably possible to use any other JDBC-compatible database

Modifier (in)compatibility

The FE_Wiki_Inc feature extractor uses information from the Wikipedia category/graph structure as described above, as well as Wordnet (also see above) to automatically compute the compatibility between the pronominal modifiers of the anaphora and antecedent - if they have matching head nouns.

Attributes and relations are extracted from the markable string of each mention, for example American tourist in Cuba would have associated with it American as an attribute, and from Cuba as a relation, which can then be compared against Cuban tourist to determine the incompatibility of the two mentions. The Wikipedia and Wordnet evalutions are computed seperately and a final score of compatibiltity is assigned based on the two.

This feature extractor needs the Wordnet library and access to the Wikipedia category/graph structure (see the respective subsections for necessary preconditions).

Chapter 4

Extracting Lexical and Commonsense Knowledge from Wikipedia

Simone Ponzetto, Jason Smith,
Vladimir Eidelman and Massimo Poesio

In this Chapter we discuss first our methods for extracting similarity information from Wikipedia's category structure, then for extracting such information from hyperlinks and redirects, and finally how we compute the incompatibility feature.

4.1 FE_Wiki_Similarity

The feature modeling semantic similarity from Wikipedia builds upon and extends previous work on using the system of categories in Wikipedia as a semantic network for computing semantic relatedness (Strube and Ponzetto 2006). In addition, it crucially makes use of a taxonomy automatically generated from that category network (Ponzetto and Strube 2007). This allows us to provide the coreference resolution system with scores of *semantic similarity* modeling the semantic compatibility between antecedent and anaphor.

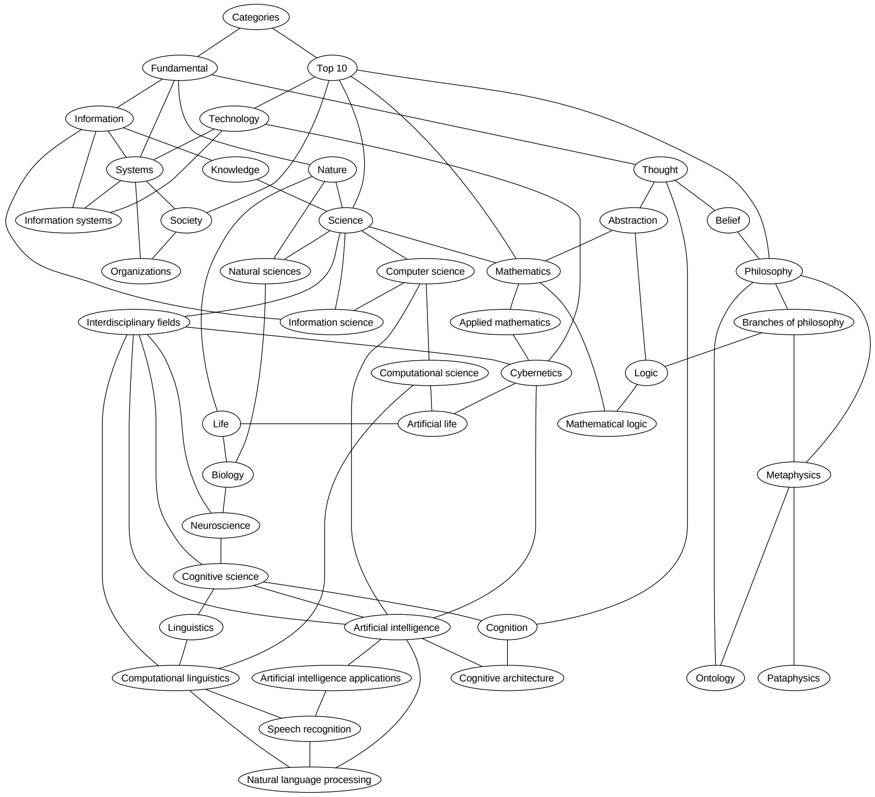


Figure 4.1: Wikipedia category network. The top nodes in the network (Categories, Fundamental, Top 10) are structurally identical to the more content bearing categories.

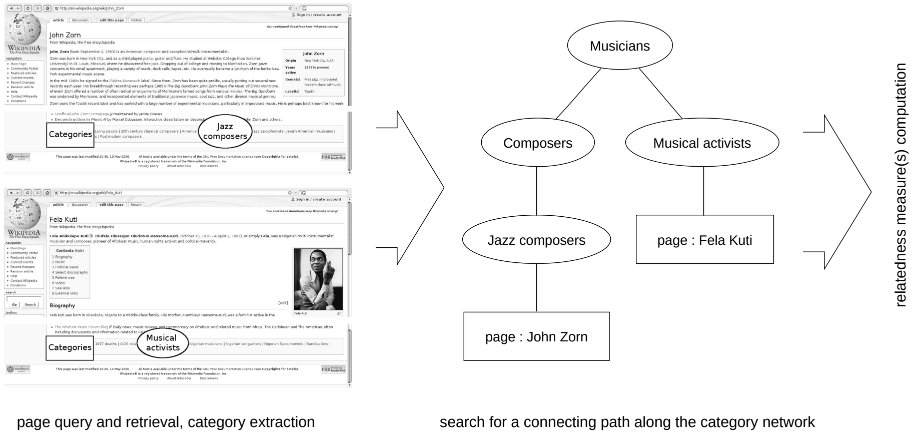


Figure 4.2: Wikipedia-based semantic relatedness computation. First, target pages for the given queries are retrieved, possibly via disambiguation. Next, categories are extracted to provide an entry point to the category network. Connecting paths are then searched along the category network using a depth-limited search. The paths found are scored and the ones satisfying the measure definitions (i.e. the shortest one for path-length measures, and the most informative one for information-content measures) are returned.

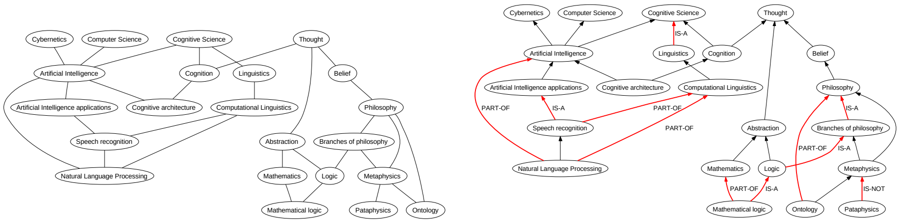
4.1.1 WikiRelate! Computing Semantic Relatedness Using Wikipedia

Strube and Ponzetto (2006) presents a method called WikiRelate! which takes the system of categories in Wikipedia as a semantic network to compute measures of semantic relatedness. Wikipedia allows in fact for structured access by means of *categories*: the encyclopedia articles can be assigned one or more categories, which are further categorized to provide a so-called "category tree" (Figure 4.1). Though not designed as a strict hierarchy, the categories form a graph which can be used as a network to compute semantic relatedness. Their work showed (1) how to retrieve Wikipedia articles from textual queries and resolve ambiguous queries based on the articles' link structure, and (2) compute semantic relatedness as a function of the articles found and the paths between them along the categorization network (Figure 4.2). For instance, given the name entities John Zorn and Fela Kuti from Figure 4.2, their semantic relatedness can be computed by finding the connecting path between their Wikipedia articles along the categorization network and using standard measures from the literature, e.g. computing semantic distance as the number of edges between pages in the hierarchy and defining semantic relatedness as the inverse score of the semantic distance (cf. Rada et al. 1989).

$$sim(c_1, c_2) = \frac{1}{\#\ nodes\ in\ path}$$
$$sim(\textsf{John Zorn, Fela Kuti}) = \frac{1}{6} = 0.16$$

Ponzetto and Strube (2006) also shows that including such scores into an NLP system dealing with coreference resolution is beneficial. A limitation of that approach is that it computes *semantic relatedness*, rather than *semantic similarity*[1]. This is because approaches to measuring semantic similarity that rely on lexical resources use paths based on *isa* relations only, whereas the Wikipedia categorization network contains relations between categories which are neither semantically typed nor show a uniform semantic. However,

[1] Semantic relatedness indicates how much two concepts are semantically distant in a network or taxonomy by using all relations between them (i.e. hyponymic/hypernymic, antonymic, meronymic and any kind of functional relations including is-made-of, is-an-attribute-of, etc.). When limited to hyponymy/hyperonymy (i.e. *isa*) relations, the measure quantifies semantic similarity instead (see Budanitsky and Hirst (2006), for a discussion of semantic relatedness vs. semantic similarity). In fact, two concepts can be related but are not necessarily similar (e.g. cars and gasoline, see Resnik (1999)).



(a) original category graph

(b) category graph augmented with semantic relations

Figure 4.3: Inducing semantic relations between categories in Wikipedia

it seems that coreference resolution needs more fine-grained semantic similarity (e.g. 'FCC *isa* an agency') rather that semantically unspecified semantic relatedness. Accordingly, in order to compute semantic similarity, one needs a subsumption hierarchy (i.e. a fully-fledged taxonomy) on top of the system of Wikipedia categories. This consists of transforming the unlabeled graph in Figure 4.3(a) into the semantic network in Figure 4.3(b), where the links between categories are augmented with a clearly defined semantics.

4.1.2 Deriving a Large Scale Taxonomy from Wikipedia

Ponzetto and Strube (2007) shows how to induce a large-scale taxonomy from the Wikipedia category network. The taxonomy consists of *isa* and *notisa* relations and it is generated using three main classes of heuristics:

syntax-based methods perform a string matching between the syntactic components of the category labels;

connectivity-based methods exploit the structure and connectivity in the network;

lexico-syntactic based methods use is-a (Hearst 1992) and part-of (Berland and Charniak 1999) surface pattern.

Using these methods, they start with a category network consisting of 165,744 categories and 349,263 direct links between them and generate a very large taxonomy, i.e. containing up to 105,418 *isa* semantic links. In order to evaluate the taxonomy, they perform an extrinsic evaluation by computing semantic similarity on two commonly used datasets, namely the list of 30 noun pairs of Miller and Charles (1991) and the 65 word synonymity list from Rubenstein and Goodenough (1965). The results show that Wikipedia-based measures of semantic similarity computed using the automatically generated taxonomy are competitive with the ones computed from WordNet (Fellbaum 1998).

4.1.3 Bringing it All Together: Computing Semantic Similarity Using Wikipedia for Coreference Resolution

The contribution of the **FE_Wiki_Similarity** feature is to capture the notion of *semantic compatibility* in terms of *Wikipedia-based semantic similarity scores* and to make this kind of information available to the coreference resolution system. In order to achieve this, it uses the original WikiRelate! method (subsection 4.1.1) on the automatically generated taxonomy (subsection 4.1.2). The taxonomy is obtained by taking only those categories in Wikipedia which are found to be in a *isa* relation and removing the top 200 categories with the highest PageRank score (Brin and Page 1998)[2], as this has been shown to yield the semantic similarity scores which best correlate with human judgments. The degree of semantic compatibility is obtained by computing the similarity scores for each input antecedent-anaphor pair as follows:

Query extraction: normalize the strings by either taking the head lemma (for common nouns, e.g. house) or the full NP (for named entities, e.g. George W. Bush).

Page retrieval: find the two Wikipedia pages $pages = \{p_1, p_2\}$ the queries refer to.

Category extraction: parse the pages and extract the two sets of categories $C_1 = \{c_1 | \, c_1 \, \text{is\_category\_of} \, p_1\}$ and $C_2 = \{c_2 | \, c_2 \, \text{is\_category\_of} \, p_2\}$ the pages are assigned to.

Path finding: compute the set of paths between *all pairs of categories*[3] of

[2] See Ponzetto and Strube (2007) for details. In a nutshell, the idea is to filter out semantically coarse-grained, over-connected categories using a link analysis algorithm (such as PageRank) by assuming that these categories are the most authoritative ones in the category network.

[3] We factorize over all possible category pairs in order to deal with the sense disambiguation problem. That is, we assume that each category captures a sense of the target query and we take the cross product of each antecedent and anaphor category to form pairs of âĂŸWikipedia synsetsâĂŹ to compute the similarity across all of them.

the two pages, namely $paths = \{path_{c_1,c_2} | \, c_1 \in C_1, c_2 \in C_2\}$.

Semantic similarity computation: compute semantic similarity using the taxonomy based on the paths found along the category network. For each measure WIKI_SIMILARITY[4], compute the similarity score for all category pairs, and create the following features:

1. **WIKI_SIMILARITY_BEST** the *highest* similarity score from all $\langle C_{RE_i,n}, C_{RE_j,m} \rangle$ category pairs.
2. **WIKI_SIMILARITY_AVG** the *average* similarity score from all $\langle C_{RE_i,n}, C_{RE_j,m} \rangle$ category pairs.

4.2 FE_Wiki_Alias

This feature takes advantage of Wikipedia's hyperlinks. The simplest form of hyperlink is created by surrounding a word or group of words with square brackets. For example, the wikitext

```
On Monday, [[Bill Clinton]] released a statement...
```

will contain a link to the article for "Bill Clinton". It is often the case, however, that a sentence will not contain the full name of an article that the author wants to link to. This type of hyperlink can be accomplished through "piped links". A piped link contains the article the text links to followed by the visible text of the link (also referred to as the alias), separated by the pipe character. For example, the wikitext

```
...since the [[Bill Clinton|Clinton]] Administration.
```

will appear as

```
...since the Clinton Administration.
```

where "Clinton" links to the article for "Bill Clinton". While this concept is very simple, it gives us several examples of alternate names for entities, including abbreviations and shortened names.

The initial version of this feature takes advantage of these piped links by firing whenever two strings link to the same article. Essentially, we are

[4] We use the path length based measures from Rada et al. (1989), Wu and Palmer (1994) and Leacock and Chodorow (1998) and the information content based measure from Seco et al. (2004).

Alias	Pages		
America	United States	Neotropics	Americas ...
USA	United States	Florida	USSF ...
British	British	England	United States ...

Table 4.1: Wiki_Alias Table

computing "bag of articles" similarity for strings, where each string has a vector of articles that it links to. This feature's value can be computed efficiently by first building a database that contains every article that each string links to. It is assumed that every string links to its own article. The table containing this data has two fields, "alias", which is the visible text of a link, and "page ids", which is a list of every page that the string in "alias" links to. Some example entries are given in Table 4.1.

There are also some problematic examples of piped links, as seen in Table 4.1. The string "British" links to "United States" somewhere in Wikipedia. (This occurs in the article for the British children's television show *Fimbles*.)

Alias	Pages		
America	United States	Neotropics	America ...
Weights	0.1648	0.0016	0.1328 ...
USA	United States	Florida	USSF ...
Weights	0.9999	2.990e-5	0.0006 ...
British	British	England	United States ...
Weights	0.1801	0.1988	0.0010 ...

Table 4.2: Wiki_Alias Table with Weights

To account for such cases, the feature was changed from binary to real valued. The strength of a match between two strings now takes into account how often each string links to a given article. Viewing each string as a "bag of articles", we can use term frequency weighting to improve the accuracy of this feature. Table 4.2 contains the updated database table, where each article has a numeric weight associated with it calculated simply by

$$\frac{\text{number of times the string X links to article Y}}{\text{number of times the string X links to any article}}$$

Since “British” only links to “United States” once, the similarity between the two is now dramatically lowered.

Since weighting by term frequency improved performance, using inverse document frequency was considered and tested. As it turns out, this did not help. Weighting by inverse document frequency in this task would mean lowering similarity scores between two strings when they link to a “popular” article (popular in this case meaning frequently linked to). This is not desirable with our dataset; the fact that Bill Clinton’s article is frequently linked to does not decrease the chance that “President Clinton” is a possible alias for “Bill Clinton”. The use of term frequency weighting was done as a way to filter out noise, not to give a better measure of similarity between two strings. Inverse document frequency, and likely any other more complex vector-based weighting, would not help with this task.

4.2.1 FE_Wiki_Redirect

Wikipedia contains many pages which transparently redirect the user to another article. These redirect pages serve to correct spelling errors (“Untied States” redirects to “United States”), expand acronyms (“NBA” redirects to “National Basketball Association”), and generally ensure that all names which unambiguously refer to an article will lead you to that article. Redirect information is an obvious choice for a string matching feature, due to the inherent reliability of the information. For example, while “William J. Clinton” redirects to “Bill Clinton”, “William Clinton” does not, since there is another William Clinton in Wikipedia. Something even more ambiguous, such as “Bill”, would never redirect to “Bill Clinton”. There is never the issue of ambiguity as with the FE_Wiki_Alias feature.

FE_Wiki_Redirect was implemented as a boolean feature which is true when two strings redirect to the same page, and false when they do not. Again, a page is considered to redirect to itself.

4.2.2 FE_Wiki_Lists

In addition to its categories, Wikipedia maintains several list pages. These lists differ from categories in several significant ways: there is little hierarchy (aside from a few “lists of lists”), items in the lists need not have articles associated with them, and if an item X belongs to a list of Y’s, there is a strong indication that Y is a hypernym of X.

The FE_Wiki_Lists feature is motivated by these observations. This boolean feature fires for strings X and Y if X belongs to a list of Y’s, or

Name	Lists		
Frederick IX	dane		
Franz Mesmer	austrian	physician	astrolog
Jerry Holland	violinist	fiddler	

Table 4.3: Wiki_Lists Table

vice-versa. For example, the page "List of cities in Bosnia and Herzegovina" contains "Bijelina", so this feature would fire on the strings "city" and "Bijelina". Naturally, some preprocessing must be done to accomplish this. First, the head word is extracted from the list's title ("cities" in this example). Next, this head word is stemmed. This data is stored in a database; each entry contains a list item followed by each list it belongs to (Table 4.3 contains some example entries).

4.3 The incompatibility feature

Attempts at incorporating semantic relatedness information automatically extracted from various sources have been made with various results. While computing semantic relatedness may seem similar to computing compatibility, thus lending itself to well studied approaches, the tasks are actually quite distinct. For semantic relatedness, graph metrics are often used that attempt to approximate the relatedness of two concepts with their distance in a graph constructed from a given knowledge source. A variety of path metrics have been used, however, none are fit for evaluating compatibility. For relatedness, path metrics are a reasonable approximation due to the fact that closely related concepts usually appear close to each other in a lexical or semantic database, while distance concepts are analogously distant in the graph. This convenient relation does not hold for compatibility.

For instance, the concepts **yellow** and **violet** are one node apart in the Wikipedia graph, indicating close semantic relatedness, however, they are perfectly incompatibility, since the yellow car would never be assumed coreferent with the violet car by a human. Compare tiny and small, they are most likely compatible, but they have a high relatedness, as opposed to smart and bony, whose relatedness is very low, but compatibility reasonably unquestionable.

We developed a system for the automatically evaluation of compatibility between modifiers. To the best of our knowledge, this is one of the first recent attempts to incorporate lexical and semantic knowledge, in the form of WordNet and Wikipedia to the task of identifying the compatibility of noun modifiers.

Compatibility assessment may be performed for all pairs of mentions compared in the course of creating coreference chains. This would allow for a greater cache of words and thus for the identification of a wider range of modifiers. However, due to practical constraint of computation time for each mention pair, coupled with the number of mentions, currently compatibility assessment is only performed when both mentions share the same head noun.

The incompatibility feature uses the fact that each mention is associated in BART with a **discourse entity** (Poesio and Kabadjov 2004), associated with a **semantic tree** containing information about the mention, such as type, location, and name. For our purposes, we only need the relations and attributes that each entity contains.

Attributes generally occur as premodifiers, as in *fast car* or *crazy doctor*, whereas relations occur as postnominals, as in *software from India*, and *snow in Vancouver*. WordNet and Wikipedia are used as outside knowledge sources.

Thus when the head noun of the antecedent matches the head noun of the anaphora, this feature fires. The compatibility of the modifiers is computed both with WordNet and Wikipedia, and the scores resulting from both sources are combined into a final feature score. We treat a score of 0 compatibility from either source as outweighing any other scores, since we deem two entities being incompatible as a stronger claim than either deeming them possibility compatible or compatible.

WordNet is a structured lexical database. Here we exploit its concept of **synsets** - groups of words with considered semantically equivalent. When computing the compatibility, we first lookup the modifiers synset and antonym set. We assign a unique id, ID, to this set, as well as -1*ID to the antonym set. These words are cached using a hashing function that allows for fast retrieval and evaluation using the unique ID, since using the word as a key, we can retrieve its unique ID and thus all the words which are both synonymous and antonymous with it.

Then we compare all the modifiers of the entities and either classify them as being perfectly incompatible (0), perfectly compatible (1), or possibly compatible (0.5). If there are multiple modifiers, this comparison can get tricky, as it would be for comparing a large green obtuse melon with a tiny jade colored melon. Each phrase contains a synonym for green, thus having a compatibility measure of 1, but deviate with tiny and large, having compatibility 0. We use an algorithm to combine these varying measures due to multiple modifiers. The greatest weight is placed on 0 compatibility, followed by .5, and finally if only 1(s) were encountered, a 1.

Wikipedia is an open encyclopedia which contains articles categorized by users. We used a graph constructed from the Wikipedia category structure for computing compatibility of two entries in Wikipedia. As mentioned earlier, path metrics do not work, however, another simple algorithm is used. Two mentions are considered compatible if one is in a hyponomy relation with the other, ie. there is a straight path from one entity node to the other, where the highest point in the path according to the Wikipedia graph structure corresponds to one of the mention nodes. Otherwise, two mentions are considered incompatible if there is an intermediary node along the path at which direction changes âĂŞ the top node according to the category structure is inside the path. For instance, when comparing the postmodifiers *United States* and *India*, the Wikipedia graph structure takes a path up from `United States` to `Former British Colonies` and down to `India`. Since the top node is `Former British Colonies`, and this is inside the path, the two are correctly deemed incompatible. As with the WordNet evaluation, we combine multiple modifiers with a heavier weight on 0 classification.

The use of both these sources is convenient, as they provided complementary information. WordNet is useful for adjective attributes, while relations and attributes involving nouns are better served with Wikipedia. For example, WordNet does not contain information that India is incompatible with US, whereas in Wikipedia, these two are easily determined incompatible through the graph algorithm described above. Conversely, Wikipedia cannot appreciate the incompatibility between small and large, whereas in WordNet one appears in the antonyms of the other.

So far no analysis of the system incorporating this feature has been done, so nothing can be said of the results at this time.

Chapter 5

Kernels for Coreference

Alessandro Moschitti, Xiaofeng Yang, Alan Jern, Massimo Poesio

In this chapter, we describe the advances of the state-of-the-art in co-reference resolution during the workshop from a machine learning perspective. For this purpose, we use the most advanced techniques of statistical learning theories, i.e. Support Vector Machines and Kernel Methods. The former produces one of the most accurate classification algorithm whereas the latter allow for abstract feature design in very large feature spaces. More in detail, we used polynomial kernels to generate combinations of manually designed features (Section 5.1), the subset tree kernel (Section 5.1.2) to generate innovative syntactic features (Section 5.2), word sequence kernels (Section 5.1.1) to describe the context of the corefering mentions (Section 5.2.4) and the Partial Tree Kernel (Section 5.1.2) to provide a novel and effective formulation of the aliases problem (Section 5.3).

5.1 Support Vector Machines and Kernels for Text

The main idea behind machine learning is the use of labeled examples described by means of feature vectors in a n dimensional space over the real number, i.e. $\Re^n$. The learning algorithm uses space metrics over vectors, e.g. the scalar product, to learn an abstract representation of all instances belonging to the target class.

For example in case of linear classifier, like Support Vector Machines, a hyperplane $H(\vec{x}) = \vec{w} \times \vec{x} + b = 0$, where $\vec{x}$ is the feature vector representation of a classifying object o whereas $\vec{w} \in \Re^n$ and $b \in \Re$ are parameters, learned from the training examples by applying the *Structural Risk Minimization* *principle* Vapnik (1995). The object o is mapped in $\vec{x}$ with a feature function $\phi : \mathcal{O} \rightarrow \Re^n$, where $\mathcal{O}$ is the set of the objects that we want to classify. o is categorized in the target class only if $H(\vec{x}) \geq 0$.

The kernel trick allows us to rewrite the decision hyperplane as:

$$H(\vec{x}) = \Big(\sum_{i=1..l} y_i \alpha_i \vec{x}_i \Big) \cdot \vec{x} + b = \sum_{i=1..l} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b.$$

where, y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in \Re$ with $\alpha_i \geq 0$, o_i $\forall i \in \{1,..,l\}$ are the training instances and the product $K(o_i, o) = \langle \phi(o_i) \cdot \phi(o) \rangle$ is the kernel function associated with the mapping ϕ.

Note that, we do not need to apply the mapping ϕ, we can use $K(o_i, o)$ directly. This allows us, under the Mercer's conditions Shawe-Taylor and Cristianini (2004), to define abstract kernel functions which generate implicit feature spaces. An interesting example is given by the polynomial kernel:

$$PK(o_1, o_2) = (c + \vec{x}_1 \cdot \vec{x}_2)^d, \tag{5.1}$$

where c is a constant and d is the degree of the polynomial. This kernel generates the space of all conjunctions of feature groups up to d elements.

5.1.1 String Kernels

Kernel functions can be applied also to discrete space to count the number of substrings that are shared by two text fragments.

Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For string s and t we denote by $|s|$ the length of the string $s = s_1, .., s_{|s|}$, and by st the string obtained by concatenating the string s and t. The string $s[i:j]$ is the substring $s_i, .., s_j$ of s. We say that u is a subsequence of s, if there exist indices $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, for $j = 1, ..., |u|$, or $u = s[\vec{I}]$ for short. The length $l(\vec{I})$ of the subsequence in s is $i_{|u|} - i_i + 1$. We denote by Σ^* the set of all string

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

We now define the feature space, $F = \{u_1, u_2..\} = \Sigma^*$, i.e. the space of all possible substrings. We map a string s in $\mathbb{R}^\infty$ space as follows:

$$\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})} \tag{5.2}$$

for some $\lambda \leq 1$. These features measure the number of occurrences of subsequences in the string s weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to their frequency of occurrences and lengths, i.e.

$$SK(s,t) = \sum_{u \in \Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{J})} =$$

$$= \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{I})+l(\vec{J})} \qquad (5.3)$$

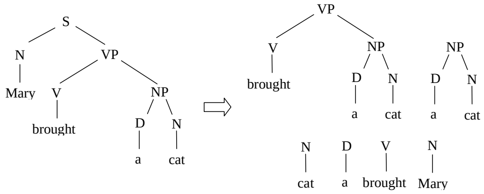
It is worth to note that if the set of symbol is defined over words a string corresponds to a word sequence and the substring space contains the set of word sequences, e.g. *George Bush goes in Iraq* contains the subsequences *George Bush*, *George Bush goes* but also *George Iraq*.

5.1.2 Tree Kernels

The kernels represent trees in terms of their substructures (fragments). The kernel function detects if a tree subpart (common to both trees) belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. We consider three important characterizations: the subtrees (STs), the subset trees (SSTs) and a new tree class, i.e. the partial trees (PTs).

We define as a **subtree** (ST) any node of a tree along with all its descendants. For example, Figure 5.1 shows the parse tree of the sentence `"Mary brought a cat"` together with its 6 STs. A **subset tree** (SST) is a more general structure since its leaves can be non-terminal symbols.

For example, Figure 5.2 shows 10 SSTs (out of 17) of the subtree of Figure 5.1 rooted in `VP`. The SSTs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an SST which has two non-terminal symbols, `V` and `NP`, as leaves whereas `[VP [V]]` is not an SST. If we relax the constraint over the SSTs, we obtain a more general form of substructures that we call **partial trees** (PTs). These can be generated by the application of partial production rules of the grammar, consequently `[VP [V]]` and `[VP [NP]]` are valid PTs. Figure 5.3 shows that the number of PTs derived from the same tree as before is still higher (i.e. 30 PTs). These different substructure numbers provide an intuitive quantification of the different information levels among the tree-based representations.



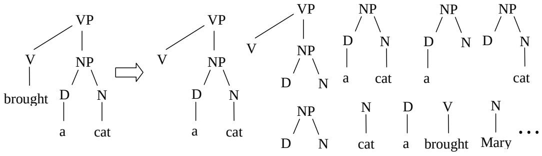


Figure 5.1: A syntactic parse tree with its subtrees (STs).

Figure 5.2: A tree with some of its subset trees (SSTs).

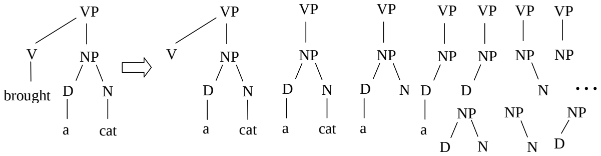


Figure 5.3: A tree with some of its partial trees (PTs).

The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. In the following the equation for the efficient evaluation of ST, SST and PT kernels are reported.

To evaluate the above kernels between two T_1 and T_2, we need to define a set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, i.e. a tree fragment space and an indicator function $I_i(n)$, equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \tag{5.4}$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. This latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes.

The Δ function depends on the type fragments that we consider as *basic* features. For example, to evaluate the fragments of type ST or SST, it can be defined as:

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminals symbols) then $\Delta(n_1, n_2) = 1$;

3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c^j_{n_1}, c^j_{n_2})) \tag{5.5}$$

where $\sigma \in \{0, 1\}$, $nc(n_1)$ is the number of the children of n_1 and c^j_n is the j-th child of the node n. Note that, since the productions are the same, $nc(n_1) = nc(n_2)$.

When $\sigma = 0$, $\Delta(n_1, n_2)$ is equal 1 only if $\forall j \quad \Delta(c^j_{n_1}, c^j_{n_2}) = 1$, i.e. all the productions associated with the children are identical. By recursively applying this property, it follows that the subtrees in n_1 and n_2 are identical. Thus, Eq. 5.4 evaluates the subtree (ST) kernel. When $\sigma = 1$, $\Delta(n_1, n_2)$ evaluates the number of SSTs common to n_1 and n_2 as proved in Collins and Duffy (2002).

Moreover, a decay factor λ can be added by modifying steps (2) and (3) as follows[1]:

2. $\Delta(n_1, n_2) = \lambda$,

3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c^j_{n_1}, c^j_{n_2}))$.

The computational complexity of Eq. 5.4 is $O(|N_{T_1}| \times |N_{T_2}|)$ but as shown in Moschitti (2006b), the average running time is linear, i.e. $O(|N_{T_1}| + |N_{T_2}|)$.

A more general form of fragments has been given in Moschitti (2006a). In this case any portion p of T, namely Partial Trees (PT), is considered and an efficient evaluation is provided. To compute it, we need to define a different Δ function:

- if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
- else

$$\Delta(n_1, n_2) = 1 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}]) \tag{5.6}$$

[1] To have a similarity score between 0 and 1, we also apply the normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{TK(T_1, T_2)}{\sqrt{TK(T_1, T_1) \times TK(T_2, T_2)}}$.

where $\vec{J_1} = \langle J_{11}, J_{12}, J_{13}, ..\rangle$ and $\vec{J_2} = \langle J_{21}, J_{22}, J_{23}, ..\rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\vec{J}_{1i}$ and $\vec{J}_{2i}$ point to the i-th child in the corresponding sequence, and $l(\cdot)$ returns the sequence length.

Furthermore, we add two decay factors: μ for the height of the tree and λ for the length of the child sequences. It follows that

$$\Delta(n_1, n_2) = \mu\Big(\lambda^2 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1)=l(\vec{J}_2)} \lambda^{d(\vec{J}_1)+d(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])\Big)\ , \tag{5.7}$$

where $d(\vec{J_1}) = \vec{J}_{1l(\vec{J}_1)} - \vec{J}_{11}$ and $d(\vec{J_2}) = \vec{J}_{2l(\vec{J}_2)} - \vec{J}_{21}$. In this way, we penalize both larger trees and subtrees built on child subsequences that contain gaps.

Equation 5.7 is a more general one, the kernel can be applied to PTs. Also note that if we only consider the contribution of the longest child sequence from node pairs that have the same children, we implement the SST kernel. For the ST computation, we also need to remove the λ^2 term from Eq. 5.7.

5.2 Kernels for Coreference Resolution

Syntactic knowledge plays an important role in co-reference resolution. Especially, the resolution of pronominal anaphora heavily relies on the syntactic information and relationships between the anaphor and of the antecedent candidates. For a practical co-reference resolution system, the syntactic knowledge usually comes from parse trees of the text. The issue that arises is how to effectively incorporate the syntactic information embedded in parse trees to help resolution. One common solution seen in previous work is to define a set of features that represent particular syntactic knowledge, such as the grammatical role of the antecedent candidates, the governing relations between the candidate and the pronoun, and so on. These features are calculated by mining parse trees, and then could be used for resolution by using manually designed rules Lappin and Leass (1994a); Kennedy and Boguraev (1996), or using machine-learning methods (Aone and Bennett 1995a; Yang et al. 2004a; Luo and Zitouni 2005).

However, such a solution has its limitation. The syntactic features have to be selected and defined manually, usually by linguistic intuition. Unfortunately, what kinds of syntactic information are effective for pronoun resolution still remains an open question in this research community. The heuristically selected feature set may be insufficient to represent all the information necessary for pronoun resolution contained in parse trees. Another problem is that the values of the syntactic features are extracted from parse trees, and the accuracy of the decoding therefore cannot be guaranteed especially for texts with complicated grammar structures.

In our study, we explore how to utilize the syntactic parse trees to help learning-based coreference resolution. Specifically, we directly utilize parse trees as a structured feature, and then use a kernel-based method to automatically mine the knowledge embedded in parse trees. The structured syntactic feature, together with other normal features, is incorporated in a trainable model based on Support Vector Machine (SVM) (Vapnik 1995) to learn the decision classifier for resolution. Indeed, using kernel methods to mine structural knowledge has shown success in some NLP applications like parsing (Collins and Duffy 2002; Moschitti 2004a) and relation extraction (Zelenko et al. 2003; Zhao and Grishman 2005). So far, there is a few work that applies such a technique to reference resolution (Yang et al. 2006b; Iida et al. 2006). But most of them focus on pronoun resolution, and to our knowledge, no work is on the coreference resolution task.

Compared with previous work, our approach has several advantages: (1) The approach utilizes parse trees as a structured feature, which avoids the efforts of decoding parse trees into a set of syntactic features in a heuristic manner. (2) The approach is able to put together the structured feature and the normal flat features in a trainable model, which allows different types of information to be considered in combination for both learning and resolution. (3) The approach is applicable for a practical coreference resolution system, as the syntactic information can be automatically obtained from machine-generated parse trees. And our study shows that the approach works well under the commonly available parsers.

We evaluate our approach in the newswire domain, on the MUC and the ACE data set. The experimental results indicate that the structured syntactic feature incorporated with kernels can significantly improve the resolution performance. Especially for the resolution of pronoun, the approach brings up to 8% in F-measure.

5.2.1 Related Work

One of the early work on pronoun resolution relying on parse trees was proposed by Hobbs (1978a). For a pronoun to be resolved, Hobbs' algorithm works by searching parse trees of the current text. Specifically, the algorithm processes one sentence at a time, using a left-to-right breadth-first searching strategy. It first checks the current sentence where the pronoun occurs. The first NP that satisfies constraints, like number and gender agreements, would be selected as the antecedent. If the antecedent is not found in the current sentence, the algorithm would traverse the trees of previous sentences in the text. As the searching processing is completely done on parse trees, the performance of the algorithm would rely heavily on the accuracy of the parsing results.

Lappin and Leass (1994a) reported a pronoun resolution algorithm which uses the syntactic representation output by McCord's Slot Grammar parser. A set of salience measures (e.g. *Subject*, *Object* or *Accusative* emphasis) is derived from the syntactic structure. The candidate with the highest salience score would be selected as the antecedent. In their algorithm, the weights of salience measures have to be assigned manually.

Luo and Zitouni (2005) proposed a coreference resolution approach which also explores the information from the syntactic parse trees. Different from Lappin and Leass (1994a)'s algorithm, they employed a maximum entropy based model to automatically compute the importance (in terms of weights) of the features extracted from the trees. In their work, the selection of their features is mainly inspired by the government and binding theory, aiming to capture the c-command relationships between the pronoun and its antecedent candidate. By contrast, our approach simply utilizes parse trees as a structured feature, and lets the learning algorithm discover all possible embedded information that is necessary for pronoun resolution.

Our previous work (Yang et al. 2006b) systematically explored the utility of the structured syntactic features for pronoun resolution. Iida et al. (2006) also tried a similar strategy on Japanese zero-anaphora resolution, but using dependency tree instead of syntactic parse tree as in (Yang et al. 2006b).

5.2.2 The Resolution Framework

Our coreference resolution system adopts the common learning-based framework similar to those by Soon et al. (2001a) and Ng and Cardie (2002a).

In the learning framework, a training or testing instance is formed by a possible anaphor and one of its antecedent candidate. During training, for each anaphor encountered, a positive instance is created by paring the anaphor and its closest antecedent. Also a set of negative instances is formed by paring the anaphor with each of the non-coreferential candidates. Based on the training instances, a binary classifier is generated using a particular learning algorithm. During resolution, an encountered noun phrase to be resolved is paired in turn with each preceding antecedent candidate to form a testing instance. This instance is presented to the classifier which then returns a class label with a confidence value indicating the likelihood that the candidate is the antecedent. The candidate with the highest confidence value will be selected as the antecedent of the possible anaphor.

As with many other learning-based approaches, the knowledge for the reference determination is represented as a set of features associated with the training or test instances. In our baseline system, the features adopted include lexical property, morphologic type, distance, salience, parallelism, grammatical role and so on. These features usually have a binary value. To distinguish with the feature based on parse tree, we call them "flat" features throughout the remaining report. Interested readers may like to refer to the other reports for the detailed description of the resolution framework of the system as well as the flat features.

5.2.3 Incorporating Structured Syntactic Information

In this section we will discuss how to represent parse tree as a structured feature and how to use the kernels to incorporate this feature to do leaning and resolution.

Main Idea

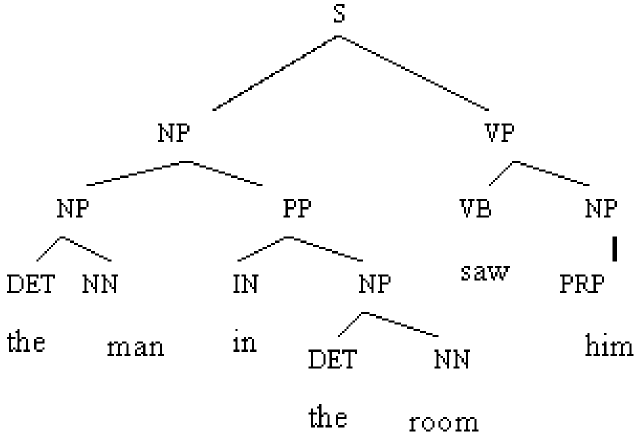
A parse tree that covers a pronoun and its antecedent candidate could provide us much syntactic information related to the pair. The commonly used syntactic knowledge co-reference resolution, such as grammatical roles or the governing relations, can be directly described by the tree structure. Other syntactic knowledge that may be helpful for resolution could also be implicitly represented in the tree. Therefore, by comparing the common substructures between two trees we can find out to what degree two trees contain similar syntactic information, which can be done using a convolution tree kernel.

The value returned from the tree kernel reflects the similarity between two instances in syntax. Such syntactic similarity can be further combined with other knowledge to compute the overall similarity between two instances, through a composite kernel. And thus a SVM classifier can be learned and then used for resolution. This is just the main idea of our approach.

Normally, parsing is done on the sentence level. However, in many cases a pronoun and an antecedent candidate do not occur in the same sentence. To present their syntactic properties and relations in a single tree structure, we construct a syntax tree for an entire text, by attaching the parse trees of all its sentences to an upper node.

For each sentence in an input text, a parse tree is automatically generated

Figure 5.4: Parse tree for instance “the man in the room saw him"



by a given parser. The trees of all different sentences are attached to a newly created root to build a large syntax tree for the entire text, based on which the structured syntactic feature is computed.

Structured Syntactic Feature

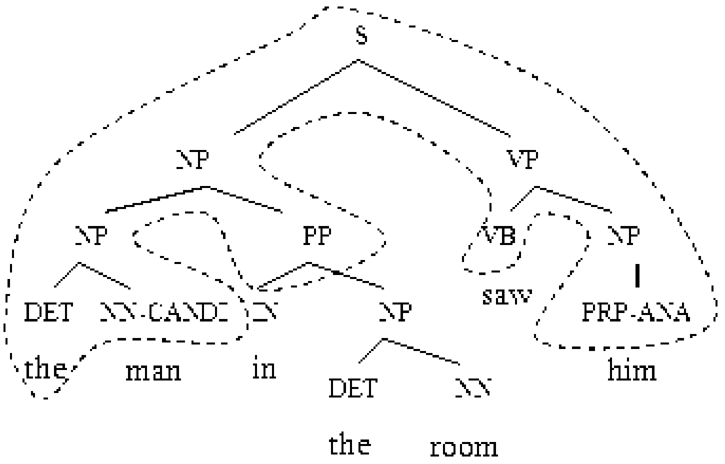
Having obtained the parse tree of a text, we shall consider how to select the appropriate portion of the tree as the structured feature for a given instance. As each instance is related to a possible anaphor and an antecedent candidate, the structured feature at least should be able to cover both of these two expressions. Generally, the more substructure of the tree is included, the more syntactic information would be provided, but at the same time the more noisy information that comes from parsing errors would likely be introduced. In our study, we consider several structured features that contain different substructures of the parse tree.

For illustration, we use the sentence “The man in the room saw him. " . The parse tree for the sentence is shown in Figure 5.4.

Consider the pronominal anaphor “him" and the antecedent candidate “The man". An instance inst(“him", “The man") is created. The following structured feature may be applied to the instance.

Tree_Ana_Candi: The feature includes the nodes occurring in the shortest path connecting the pronoun and the candidate, via the nearest commonly commanding node. Also it includes the first-level children of the

Figure 5.5: The structured feature Tree_Ana_Candi for instance inst(“the man", “him") in the sentence ‘the man in the room saw him"



nodes in the path. To reduce the data sparseness, the leaf nodes representing the words are not incorporated in the feature, except that the word is the word node of the “DET" type (this is to indicate the lexical properties of an expression, e.g., whether it is a definite, indefinite or bare NP)

If the pronoun and the candidate are not in the same sentence, we do not include the nodes denoting the sentences (i.e., “S" nodes) before the candidate or after the pronoun.

Figure 5.5 shows such a feature for the instance i“him”, ”the man”, which is highlighted with dash lines. Note that to distinguish from other words, we explicitly mark up in the structured feature the pronoun and the antecedent candidate under consideration, by appending a string tag “ANA" and “CANDI" in their respective nodes (e.g.,“NN-CANDI” for “man” and “PRP-ANA” for “him" as shown in Figure 5.5).

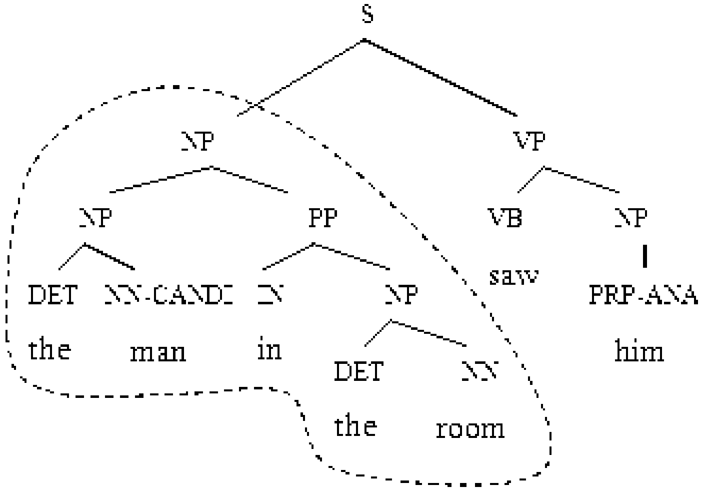
From the figure, the structured feature can disclose at least the following information:

1. The candidate is post-modified by a preposition phrase. (the node “PP” for “in the room” is included)

2. The candidate is a definite noun phrase (the article word “the" is included).

3. The candidate is in a subject position (NP-S-VP structure)

Figure 5.6: The structured feature Tree_Candi for instance inst(“the man", “him") in the sentence ‘the man in the room saw him"



4. The anaphor is an object of a verb. (the node “VB” for “saw” is included).
5. The candidate is c-commanding the anaphor (the parent of the NP node for “the main in the room" is dominating the anaphor (“him")

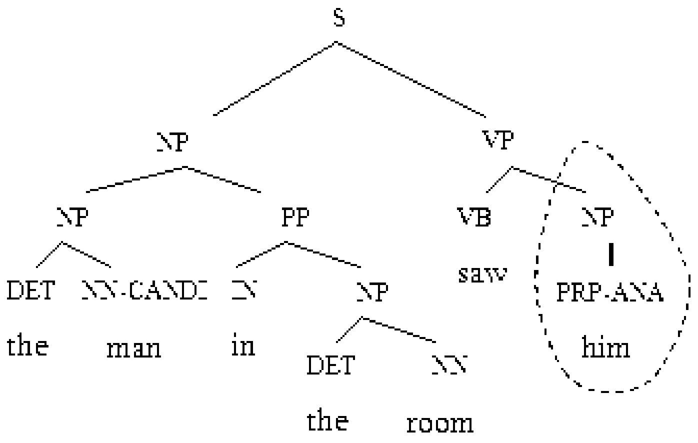
All the above information is important for reference determination in the pronoun resolution.

Tree_Candi: The feature includes the whole tree structure related to the NP of the candidate. Given a candidate, we first identify the entry of the head word (e.g., “man"). Then we trace the ancestors of the head word in the tree from bottom to top, until we get to first NP node whose parent is a non-NP node and whose right siblings are all non-NP node (i.e. the NP node for “the man in the room"). The NP node found is the most upper node covering the expression of the anaphor.

Figure 5.6 shows the feature for the sample sentence. Such a feature could represent more detailed description information of the candidate.

Tree_Ana: The feature includes the whole tree structure related to the NP of the candidate. The selection of the feature is similar to that for the feature Tree_Candi. Given an anaphor, we first identify the entry of the

Figure 5.7: The structured feature Tree_Ana for instance inst(“the man", “him") in the sentence ‘the man in the room saw him"



head word (e.g., “him"). Then we trace the ancestors of the head word in the tree from bottom to top, until we get to first NP node whose parent is a non-NP node and whose right siblings are all non-NP node (i.e. NP node for “him").

Figure 5.7 shows the feature for the sample sentence. Such a feature could represent more detailed description information of the anaphor.

5.2.4 Encoding Context via Word Sequence Kernel

The above structures aim at describing the interaction between one referential and one referent; if such interaction is observed on another mention pair, an automatic algorithm can establish if they corefer or not. This kind of information is the most useful to characterize the target problem, however, the context in which such interaction takes place is also very important. Indeed, natural language proposes many exceptions to linguistic rules and these can only be detect by looking at the context.

A typical way to encode context in NLP is regards the use of a window of k words around the target concept, e.g. the mention candidate. More sophisticated approaches use either n-grams or important co-occurrent word pairs which are not necessarily sequential, i.e. there can be some other not important words between them.

The extraction of all these features is expensive in term of program coding and the dimension of feature spaces that would result from the extraction of all possible n-grams (containing also gaps). The solution to this problem is provided by the string kernel with gap based on words (introduced in Section 5.1.1).

For example, given the following context of *Bill Gates*: *and so* ***Bill Gates*** *says that*, i.e. a window of 4 words. A string kernel would extract features likes: *Bill Gates says that*, *Gates says that*, *Bill says that*, *so Gates says that*, *and so that* and so on.

5.2.5 Experiments

In our experiments, we tested the Polynomial Kernel, Tree Kernels and Word Sequence Kernel and on the pronoun and Coreference resolution tasks on the MUC-6 and ACE03-BNews data set. Regarding Tree Kernels for syntax, we only show the results of the **Tree_Ana_Candi** structure as it was the only one producing an improvement.

A preliminary investigation of different kernel combinations on a validation set showed that

$$CK = TK(T1, T2) \cdot PK(\vec{x}_1, \vec{x}_2) + PK(\vec{x}_1, \vec{x}_2) \tag{5.8}$$

is the best combination between the tree kernel, TK, applied to the Tree_Ana_Candi structure and a polynomial kernel, PK, of degree 2, over the basic manual features.

Additionally, the word sequence kernel, SK, improved the above kernel by simply summing it to the combination, i.e. by using $CK + SK$ in the learning algorithm.

Table 5.1 lists the results for the pronoun resolution. We used PK on the Soon et al.'s features as the baseline. On MUC6, the system achieves a recall of 64.3% and precision 63.1% and an overall F-measure of 63.7%. On ACE02-BNews, the recall is lower 58.9% but the precision is higher, i.e. 68.1%, for a resulting F-measure of 63.1%. In contrast, adding the syntactic structured feature leads to a significant improvement in 17% precision for MUC-6 with a small gain (1%) in recall, whereas on the ACE data set, it also helps to increase the recall by 7%. Overall, we can see an increase in F-measure of around 8% for MUC and 4.5% for ACE02-BNews. These results suggest that the structured feature is very effective for pronoun resolution.

Table 5.2 lists the results on the coreference resolution. We note that adding the structured feature to the polynomial kernel, i.e. using the model PK+TK, improves the recall of 1.9% for MUC-6 and 1.8% for ACE-02-BNews and keeps invariate the precision. Compared to pronoun resolution,

Table 5.1: Results of the pronoun resolution

	MUC-6			ACE02-BNews		
	R	P	F	R	P	F
Base Features	64.3	63.1	63.7	58.9	68.1	63.1
Base Features+Syntactic Tree	65.2	80.1	71.9	65.6	69.7	67.6

Table 5.2: Results of the coreference resolution

	MUC-6			ACE02-BNews		
	R	P	F	R	P	F
PK	61.5	67.2	64.2	54.8	66.1	59.9
PK+TK	63.4	67.5	65.4	56.6	66.0	60.9
PK+TK+SK	64.4	67.8	66.0	57.1	65.4	61.0

the improvement of the overall F-measure is smaller (less than 1%). This occurs since the resolution of non-pronouns case does not require a massively use of syntactic knowledge as in the pronoun resolution problem. Still, the enhancement in F1 suggests that adding structured features can help in some cases, e.g., the identification of NP pairs in a predicative structure.

Finally, it should be noted the positive impact of the context information provided by the string kernel, especially in the MUC dataset, i.e. +0.6%. This shows that the classification algorithm can exploit the knowledge about different contexts to make different decisions, e.g. it can disambiguate the sense of *bank* in the two cases: *the central bank will finance new construction plans, for example, the building of* ***a river bank in the*** *city.* vs. *the central bank will finance new construction plans, for example****, the bank will finance*** *a new hospital in the city.* Although, the set of words in the two sentences are very similar the windows of 4 words around *bank* are quite different.

5.3 Kernels for Alias Resolution

Most methods currently employed by coreference resolution (CR) systems for identifying coreferent named entities, i.e. aliases, are fairly simplistic in nature, relying on simple surface features such as the edit distance between two strings representing names.

The fundamental problem with this approach is that it fails to take into account the structure contained within names. For instance, we know from experience that last names tend to be more unique than first names and therefore, the name *Jane Hauk* is slightly more likely to be coreferent with the name *Hauk* than the name *Jane* even though both names are the same edit distance away from *Jane Hauk*.

In light of this shortcoming of existing named entity resolution systems, we propose an alternative method that takes advantage of the syntactic structure of names and combines this information with the use of kernel methods, a set of more sophisticated and flexible functions for measuring similarity between two objects.

For the purposes of this paper, we focused exclusively on proper nouns, or named entities. Table 1 provides several examples of the cases we were concerned with, taken from the MUC 6 corpus (Chinchor and Sundheim 2003). Our primary goal was to determine whether improving named entity resolution using specialized features and a separate classifier could lead to an overall improvement in performance of a general CR system. In particular, we sought to exploit syntactic structure in proper names to help identify aliases. This paper describes our approach.

5.3.1 Related Work

Although several previous studies have been conducted that deal with the named entity resolution task, they are all relatively simplistic in that they either do not rely on any machine learning mechanism or do not make use of names' syntactic structures, instead treating them as simple strings.

Wee M. Soon (2001) include a basic *alias* feature in their CR architecture. The binary feature, which is set to `true` if two mentions are determined to be likely aliases, makes use of several specialized heuristics for different types of named entities. For people, only the last token in each string is compared for equality. For organizations, one mention is attempted to be made into an abbreviation of the other. This approach has the advantage of being able to

Name	**Alias**
BJ Habibie	Mr. Habibie
Federal Express	Fedex
Ju Rong Zhi	Ju

Table 5.3: Examples of coreferent named entities (aliases) taken from the MUC 6 corpus.

fairly reliably detect abbreviations, but it does not treat mentions marked as people in a very sophisticated way.

Bontcheva et al. (2002) present a more sophisticated approach in the form of a set of binary rules that collectively make up a named entity resolution module. In addition to the **last token match** rule used by Soon et al., several more rules are introduced, including:

- **possessive matching** – matches a name and its possessive form (e.g. *Greg* and *Greg's*).
- **prepositional phrase matching** – matches organization names that have been rotated around a preposition (e.g. *Department of Defense* and *Defense Department*).
- **multi-word name matching** – matches two phrases in which one name is a substring of another (e.g. *The President of the United States* and *The President*).

Several other rules are used. One shortcoming of this approach as it is used in their system is that if any single rule fires, two mentions are classified as aliases. In other words, no machine learning is employed.

Uryupina (2004) presents the most in-depth study of the subject. They split the task of matching two aliases into three stages: *normalization* – removing largely irrelevant information such as capitalization, punctuation and determiners; *substring selection* – picking the most salient token in each name for comparison; and *matching* – comparing the substrings. After developing several different features for each of these processing stages and testing a variety of feature configurations, they report that sophisticated matching algorithms can successfully improve baseline performance of named entity resolution. The use of effective normalization and substring selection algorithms also leads to marginal but statistically significant gains.

5.3.2 Method

All previous studies have treated names as simple strings. We sought to improve upon past work by adding syntactic information to the feature set by tagging the parts of a name (e.g. *first name*, *last name*, etc.) as illustrated in Figure 5.8.

One clear advantage of this approach is that, assuming information about a name's internal structure is available, the *substring selection* task becomes

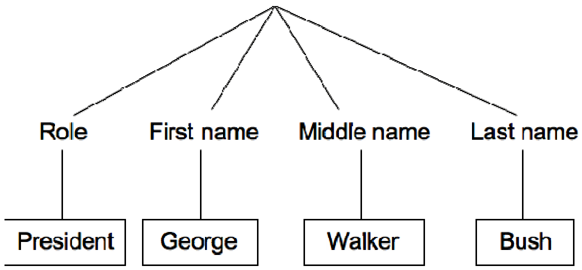


Figure 5.8: A proper name labeled with syntactic information.

straightforward for nearly all cases: simply compare the last names of two mentions.

However, a more promising advantage to this type of name representation is that it may eliminate the need for explicitly performing a *substring selection* procedure altogether, when it is used in conjunction with a machine learning algorithm. If a classifier is trained on many such examples, it is likely to assign a high weight to matching last names and a low weight to first names. In other words, the three name matching stages described by Uryupina (2004) may be replaced entirely with a learning algorithm. This is what we hoped to accomplish. This method, however, poses two significant problems: (1) how to obtain the name structure information and (2) how to represent training instances for a learner. We will address each of these problems in turn.

Name Internal Structure

Rather than build a tool for parsing proper names, we took advantage of an existing tool, developed by Hal Daumé III, called the High Accuracy Parsing of Name Internal Structure (HAPNIS) script[2].

HAPNIS takes a name as input and returns a tagged name like what is shown in Figure 5.8. It uses a series of heuristics in making its classifications based on information such as the serial positions of tokens in a name, the total number of tokens, the presence of meaningful punctuation such as periods and dashes, as well as a library of common first names which can be arbitrarily extended to any size. The tag set consists of the following: *surname*, *forename*, *middle*, *link*, *role*, and *suffix*.

[2] The script is freely available at `http://www.cs.utah.edu/ hal/HAPNIS/`.

Daumé reports a 99.1% accuracy rate on his test data set. We therefore concluded that it was sufficient for our purposes.

Tree Kernels

Once we have a collection of tagged names, the next problem is how to measure similarity between a pair of names for the purpose of training a classifier. Due to the natural representation of a name in a tree structure, we chose to follow recent successes using tree kernels (Moschitti 2004b, 2006a) with support vector machines (SVMs).

The basic principle behind a tree kernel, or more specifically a partial tree kernel, as presented in Moschitti (2006), is that two parse trees are decomposed into all their possible *partial trees*[3] and then the partial trees are compared to obtain a measure of similarity. Nodes higher in a tree will appear in more partial trees and will therefore factor more significantly into the similarity computation.

However, using partial tree kernels in this manner will only provide a similarity measure between parse trees which may then be used as a feature in a learning algorithm. A learning algorithm that uses a tree kernel as its only feature will essentially only be determining an optimal threshold value for separating positive and negative instances. Instead, we would like the learning algorithm to learn deeper characteristic differences between the training instances themselves. That is, the learning algorithm should learn what qualities of a training instance make it a positive (coreferent) or negative instance.

We therefore developed the following method of collapsing two names into a single tree representation, illustrated in Figure 5.9. This representation roughly reflects how similar two names are, but leaves the task of determining which elements of the tree are relevant for classification up to the learning algorithm. Note that this representation is essentially identical to the tree representation of names shown in Figure 5.8, but with numerical values at the leaf nodes instead of strings. These numerical values constitute a similarity measure between the corresponding parts of the two names in the training instance.

We used a string kernel function to compute the similarity between parts

[3] A *partial tree* is simply a subtree without any constraint that the subtree satisfy any rules of grammar. In other words, a set of partial trees is obtained by ignoring the semantics of a parse tree and extracting all its subtrees. The term *subtree*, however, already has a meaning in this context, namely, one in which the grammatical constraint holds.

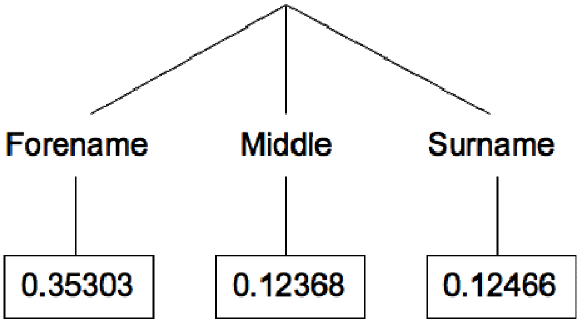


Figure 5.9: A tree representing a training instance for the names *Gregor Johann Mendel* and *Charles Robert Darwin*. The actual training instances contained branches for all of the possible HAPNIS name tags.

of names. String kernels work in essentially the same way as tree kernels, by extracting all of substrings of a string and comparing them with all the substrings of another string (Lodhi et al. 2002). One important property of string kernels is that the set of substrings it considers includes substrings of contiguous *and* noncontiguous letters. For example, the set of substrings for the string *bug* is: $\{bug, bu, bg, ug, b, u, g\}$. This approach is far superior to a simple string match for two reasons. First, it offers a graded measure of similarity instead of a rigid binary measure that a string match feature would provide. Second, it is not nearly as sensitive to alternative spellings or misspellings of names (e.g. *Hezbollah* and *Hizballah*). In a test described below, the string kernel function was also found to be more reliable for classifying aliases than the Levenshtein edit distance metric.

A tree kernel function ordinarily expects strings at the leaf nodes and, in fact, performs a string kernel when comparing leaf nodes to obtain a similarity measure. Because we used numerical values for leaf nodes, we modified the tree kernel function to take the product of these values when comparing leaf nodes to obtain some measure of their mutual similarity.

5.3.3 Data

We used the MUC 6 coreference corpus for all experiments. For our preliminary experiments, we extracted only those pairs in the MUC 6 testing set in which both mentions were proper names, as determined by the named entity recognizer built in to the general CR system we worked with. This set of proper names contained about 37,000 pairs of which about 600 were positive

Feature	**Recall**	**Precision**	**F-measure**
String kernel	49.5%	60.8%	**54.6%**
Edit distance	23.9%	53.1%	33.0%

Table 5.4: Comparison between string kernels and edit distance as predictors of aliases.

instances. About 5,500 pairs were randomly selected as test instances and the rest were used for training.

For the final experiment involving the full CR system, we used the complete MUC 6 corpus.

5.3.4 Experiments

Preliminary Experiments

As explained earlier, we chose to use a string kernel function to obtain numerical measures of similarity between corresponding parts of names instead of Levenshtein edit distance. This decision was based on an experiment in which we trained a decision tree classifer with the smaller data set of only proper name pairs, using either the string kernel of the two names or the edit distance of the two names as the only feature. The results appear in Table 2. String kernels performed better by a measure of 21.6 percentage points in the F-measure.

The second preliminary experiment we ran was to find whether using the tree-based feature described above could improve beyond the performance of using only a string kernel feature. First, an SVM classifier was trained using only a string kernel feature, just as in the previous experiment. Then an SVM classifier was trained using only the tree-based feature. The results of the comparison are shown in Table 3. Different figures were obtained for the string kernel feature because an SVM classifier was used instead of a decision tree. We did, however, use the same data set. The tree-based feature that took name internal structure into account led to notably better performance than just the string kernels, improving both recall and precision.

Final Experiment

The goal of our final experiment was to incorporate our new feature into a full CR system to see if the improvement transferred to a more comprehensive

Feature	**Recall**	**Precision**	**F-measure**
String kernel	58.4%	67.5%	62.6%
Tree-based feature	64.8%	70.0%	**67.3%**

Table 5.5: Comparison between string kernels and tree-based feature using name internal structure information.

Features	**Recall**	**Precision**	**F-measure**
Soon et al.	43.6%	74.8%	55.0%
+ Tree-based feature	43.8%	75.2%	**55.4%**

Table 5.6: The effect on coreference resolution performance of adding the tree-based feature to the existing basic set of features from Soon et al. (2001)

scenario.

We started with a basic implementation of the Wee M. Soon (2001) system. We modified it slightly to employ two different classifiers, rather than just one. One classifier was used only for instances involving two proper names and the other classifier was used for all other cases. Furthermore, the former classifier was modified to use an SVM rather than a decision tree. The reason for this is simply that our new tree-based feature required the use of an SVM.

The system was first run with both classifiers training on the exact same basic set of features described in Soon et al. Next, the classifier for proper names was modified to include the tree-based feature. The results are shown in Table 4. Use of the tree-based feature marginally improved overall CR performance.

5.3.5 Discussion and Conclusion

Although the inclusion of this new feature for identifying aliases on the basis of the similarity of the internal structure of names only had a marginal positive effect on overall performance, we consider this a promising result for the approach we described. There are several elements of our method that could be improved and refined that may lead to more significant performance gains.

First, as explained earlier, we chose to use the HAPNIS tool for extracting structure information in names. This tool, though it does seem adequate for most practical conditions, it has a few shortcomings. For instance, its behavior is based only on a handful of fixed rules, and is not the product of extensive training with a machine learning model. Thus, we are likely to see the greatest change in performance by refining this stage of the process and developing a more sophisticated tool for the job.

Second, we only tried representing the instances such that there is just one branch extending from the root for each part of a name. This method does not fully take advantage of the tree kernel function, which is most effective when there are multiple levels in a tree. Therefore, an alternative approach might involve a more complex tree representation. For instance, because the last name is undoubtedly the best indicator of whether two names are coreferent, it may make sense to place that branch higher in the tree and, perhaps, make the rest of the parts of the name children of the last name node.

Nevertheless, we have demonstrated that internal name structure is useful for named entity resolution and our approach is feasible and promising. Furthermore, this study suggests that developing specialized features for particular types of noun phrases can be an effective technique for CR. Further work is surely needed to attempt the types of improvements just described and attempt to generalize these findings by focusing on improving other types of noun phrases.

Chapter 6

Annotation and Evaluation Metrics

Ron Artstein, David Day, Janet Hitzeman and Massimo Poesio

6.1 The ACE 2005 CDC Corpus

Since there was no other sizable cross-document coreference data set available for which well-motivated intra-document coreference annotations were also available, Janet Hitzeman and David Day created and contributed to the workshop a version of the complete English ACE2005 EDT data set annotated for cross-document coreference.

6.1.1 Callisto/EDNA Annotation Tool

In order to create the ELERFED cross-doc coreference corpus, we made use of the previously developed Callisto/EDNA annotation tool. This is a specialized annotation task plug-in for the Callisto corpus annotation tool.[1]. This Callisto client plug-in requires the installation and set up of a separate Tomcat web server and associated Lucene web services plugins created for this task. The ACE2005 source and APF (standoff annotation) data are hosted on a server and indexed using a customized Lucene document parser. The result of this process is that search engine clients can search the ACE2005 repository using specialized structure-dependent queries, such as searching for strings within entity name mentions, and/or within entities of

[1] `http://callisto.mitre.org/`

a particular type and sub-type. The Callisto annotation tool task provided an integrated interface where EDT-annotated documents can be examined and individual entities can be linked to other entities in the corpus.

6.1.2 Corpus Pre-Processing and Cross-Document Co-Reference Annotation

To make the annotation process tractable, it had been already been noted in earlier discussions within the ACE community that entities without name mentions should be avoided in the cross-document coreference resolution process. Therefore, we configured the Callisto/EDNA tool to focus the annotation process on entities that met the following criteria:

1. The entity had at least one mention of type NAME;

2. The entity was of type PER, ORG, GPE or LOC.

Our goal in providing a fully annotated ACE cross-document corpus was to proceed as quickly as possible, since we were well aware of the limited time and staffing available for this task. To expedite the annotation process, we decided to apply an initial automated pre-annotation (cross-doc linking) process prior to manual annotation. We had observed in early efforts that much of the time invested by the human annotator was in physically linking frequently occurring entities to each of the numerous entities in other documents where such entities were mentioned. For example, “the President of the United States” occurred in a significant percentage of the ACE documents, and the annotation of this phrase would necessitate a laborious process of stepping through the physical clicking (actually a whole cascade of user mouse actions) of many highly probable co-referring entities. The automatic pre-processing procedure was written in Java to load the complete ACE2005 corpus EDR annotations into memory, after which it proceeded to link each pairwise entity just in case those two entities were of exactly the same TYPE and SUB-TYPE and the entities shared at least one mention of type NAME whose strings were identical (using a case-sensitive string comparison test). Of course, this automatic linking procedure produced inappropriate links. The Callisto/EDNA annotation tool provided the ability to quickly review and, if warranted, edit any links made earlier, whether by this automatic process or by a human annotator.

After an initial joint foray into the manual cross-document annotation process by both of us, Dr. Hitzeman pursued and completed the vast majority of annotation before the workshop and during its first week. The resulting corpus (derived from the ACE2005 English EDT corpus) consists of approximately 1.5 million characters, 257,000 words and 18,000 distinct document-level entities (prior to cross-document linking), and approximately 55,000 entity mentions. The document-level entities are distributed across entity types in the following way: PER 9.7K, ORG 3K, GPE 3K, FAC 1K, LOC 897, WEA 579, VEH 571. The entity mentions are distributed across mention type in this way: PRO 20K, NAM 18K, NOM 17K. Those entities that satisfy the constraints required for them to be included in the cross-document annotation process number 7,129. After the combination of automatic and manual annotation, the number of cross-document entities numbers 3,660. Of these, 2,390 are entities that are mentioned in only one document. The main effort in annotating these data required approximately 2 staff weeks, though review and editing continued for some time into the workshop.

6.2 The Arrau Corpus

The Arrau corpus of anaphoric relations was created at the University of Essex between 2004 and 2007 as part of the Arrau project, EPSRC grant number GR/S76434/01.[2] The project introduced an annotation scheme specifically targeted at marking two phenomena which had been difficult to annotate: ambiguous expressions which may refer to more than one object from previous discourse, and expressions which refer to abstract entities such as events, actions and plans. During the 2007 summer workshop the corpus was extended, consolidated, and converted for use with the anaphora resolution system developed at the workshop.

6.2.1 Composition

The corpus consists of a mixture of genres – dialogue, narrative and newswire. Task-oriented dialogues from the Trains-91 and Trains-93 corpora (Gross et al. 1993; Heeman and Allen 1995) were marked for coreference in Essex in the summer of 2006. Narratives, including five texts from the Gnome corpus (Poesio 2004b) and the full English Pear Stories corpus (Chafe 1980)[3] were annotated in Essex in the summer of 2006 and in the spring of 2007, respectively. The final and largest part of the corpus is newswire text from the Wall Street Journal portion of the Penn Treebank (Marcus et al. 1993),

[2] `http://cswww.essex.ac.uk/Research/nle/arrau/`

[3] `http://www.pearstories.org`

Source	Texts	Markables		Words
		total	anaphoric[a]	
Trains 91	16	2874	1679	14496
Trains 93	19	2342	1327	11287
Gnome	5	6045	2101	21599
Pear stories	20	3883	2194	14059
Wall Street Journal	50	9177	2852	32771

[a] Anaphoric markables are all the ones for which an explicit nominal antecedent was identified.

Table 6.1: Composition of the Arrau corpus

annotated in stages in Essex in the summers of 2006 and 2007, and at Hopkins during the workshop. The composition of the corpus is summarized in Table 6.1.

6.2.2 Annotation Scheme

The corpus was created using the MMAX2 tool (Müller and Strube 2003), which allows marking text units at different levels. All noun phrases are treated as markables which can be anaphoric or serve as antecedents (or both), and all clauses are treated as potential antecedents for discourse anaphora. For those texts where NPs and clauses were not already marked we identified them using the Charniak parser (Charniak 2000) and then corrected the output by hand. The scheme allows for marking a small number of attributes on each NP – gender, grammatical function, number, person, and an attribute which combines animacy and a concrete/abstract distinction.

Each noun phrase is marked as either anaphoric, discourse-new, or non-referential. If an object is referential then the referent is identified – in a restricted domain like Trains the referent is selected from a list, and otherwise it is entered as free text. Expressions which are anaphoric are linked to previous discourse. In order to allow the marking of expressions with ambiguous antecedents, anaphoric links are marked by pointers from an anaphoric expression to its antecedent; ambiguity is indicated by multiple pointers from a single anaphoric expression (Poesio and Artstein 2005). Anaphora is therefore not an equivalence relation and markables form more complex structures than equivalence sets indicating identity of reference. Reference to an event, action or plan is marked by a pointer from the referring NP to the clause that introduces the abstract entity (Artstein and Poesio 2006). The scheme also allows the marking of certain bridging relations, namely part-of, set membership, and a converse relation (when an expression has a *different* referent than a preceding expression).

6.2.3 Using the corpus

The anaphora resolution system developed at the workshop treats anaphoric reference as an equivalence relation, and consequesntly requires both training and test data which divide the markables into equivalence classes. In order to run the system on the Arrau corpus we created a new annotation level of markable sets, which included all the anaphors and their antecedents. The markable sets were derived from the original markable pointers, and for ambiguous anaphors we just chose the first marked interpretation, assuming that this would be the most salient one. We also augmented the Wall street Journal part of the corpus with additional texts from the Vieira and Poesio Corpus (Poesio and Vieira 1998b) and the Moscow corpus being created by Prof. Kibrik and his group at the University of Moscow.[4] The extended corpus was divided into development, training and test sets, with the test texts all taken from section 23 of the Penn Treebank. The composition of the extended corpus is shown in Table 6.2.

In the course of the workshop we only got to use the Arrau corpus with the baseline system. Performance was at around 0.40 (MUC F-score), give or take a few percentage points, regardless of which sets were used for training and testing. We plan to run the improved systems on the corpus, and release the corpus for general use.

6.3 Co-Reference Resolution Scoring Metrics

6.3.1 Existing metrics

A variety of metrics have been used to evaluate IDC, though the predominant metric in the literature is that of the MUC-6 so-called “model theoretic” coreference metric (Vilain et al. 1995). This metric tends to produce relatively high scores compared to other available metrics, but continues to be used to compare systems, so the workshop remained committed to providing performance measures in those terms.

[4]

Source	Texts	Markables		Words
		total[a]	coreferent[b]	
Arrau development	3	756	344	2593
Arrau testing	16	3289	1619	11760
Arrau training	31	5132	1874	18418
VPC develoment	15	2471	807	9900
VPC training	20	5624	2037	21218
Moscow testing	6		213	2295
Moscow training	34		1901	20234
Total testing	22		1832	14055
Total training[c]	83		5637	57557

[a] The anaphora resolution system only uses coreferent markables, so we did not extract all the markables from the Moscow corpus.

[b] Coreferent markables are those which participate in an anaphoric chain as either anaphor or antecedent.

[c] The total training data is less than the sum of the individual components because two texts are annotated in both Arrau and the VPC.

Table 6.2: Extended Arrau corpus

The MUC-6 model-theoretic metric captures the sets of mentions derived from intersections between the system-generated sets and the reference sets. By defining the theoretical lower-limit of operations required to bring the system set into conformance with the reference set, the metric defines a counterpart to the systemâĂŹs recall score. Conversely, one can define the lower bound on operations that would bring the reference set into conformance with the system set as the systemâĂŹs precision score. These two scores are then combined using the F-measure harmonic mean to derive the overall MUC-6 coreference score. Among the criticisms applied to the MUC-6 metric is that it fails to distinguish between the number of elements in the system/reference intersection sets, but assigns equal weight to the operations required to bring them back into conformance. Thus, if a system inappropriately links two very long chains and correctly separates out a short coreference chain, this generates the same scalar value as when a system links one of the long chains to the much shorter chain. Intuitively there are more pairs of mentions that âĂIJin the wrong coreference relationshipâĂİ in the first situation compared to the second, but the MUC-6 scorer treats the single âĂIJde-linkingâĂİ operation as the âĂIJcostâĂİ of their being out of alignment. In addition, the way in which recall and precision are computed completely separately allows the MUC-6 scorer to implicitly perform conflicting operations in order to compute the degree of mis-alignment (number of operations required to bring system and reference into conformance). In this sense the metric can be viewed as overly âĂIJoptimisticâĂİ in its assignment of the cost of an inappropriate link. Finally, another criticism leveled against the MUC-6 metric is that it is not defined for singleton mention sets âĂŞ if either the reference or the system coreference chains include chains of single mentions, these are simply ignored by the computation. Clearly in the case of cross-document coreference it is highly likely that entities might be mentioned in only a single document from a set.

B-Cubed (Bagga and Baldwin 1998b) is a metric designed to the MUC-6 scorerâĂŹs inattention to the number of mentions within an intersection set. In addition, this metric was specifically designed by its authors to be applicable to the task of evaluating cross-document coreference. It adopts the identical model-theoretic view of the system and reference mention sets, but, rather than summing the number of operations needed to bring each into conformance with the other, B-Cubed computed a weighted score for each intersection sub-set (in the recall and precision scenarios) based on the number of mentions in/out of the same set membership. The result is greater fidelity in distinguishing links that bring together chains of different lengths. The metric is also defined for handling coreference chains that consist of singleton mentions. In general the B-Cubed scores tend to be somewhat lower than those of the MUC-6 metric.

The ACE Value metric (Doddington 2001) was introduced in the ACE community evaluations. It computes a score based on a particular mapping between system entities and reference entities. Entities that fail to be mapped are assigned a cost, and the quality of mapped entities is a function of the number of mentions in common, as well as other features associated with the entities (their type, sub-type, mention-type, etc.). The ACE scorer generates the best possible score for a given system output by conducting a dynamic programming search of all possible mappings. The final ACE Cost Value is reported as a percentage of the possible (optimal) value (derived by mapping reference to itself) given the systemâĂŹs mapping value. Given the cost values assigned to false alarms, the ACE metric can take on negative values. The ACE value makes use of a cost matrix that treats different types of mentions and entities differently, based on what was studied as of value to real intelligence analysts using the results of a hypothetical ACE-type extraction system. The complexity of the scorer, including its cost matrix, has tended to reduce the use of this metric in reviewed publications, outside those developed specifically for the ACE community evaluations. The perl script authored by George Doddington, and distributed via the ACE web site, is the only known complete and official implementation of this scoring metric.

Xiaoqiang Luo of IBM has developed an alternative coreference scoring metric that incorporates much of the approach found in the ACE scorer. LuoâĂŹs **Constrained Entity-Alignment F-measure** (Luo 2005) also conducts an optimal mapping of system to reference mention chains that uses dynamic programming to perform this search efficiently. Unlike the ACE metric, it does not make use of a mention-specific cost matrix. It computes an analogue to recall and precision by measuring the alignment scores as a function of the system to the reference (ignoring the reference or system entities that are not mapped in these respective conditions). So far as we know LuoâĂŹs implementation of his scorer is the only one available.

Researchers at the University of Massachusetts have begun using a metric they call simply **pairwise**. It generates a simple recall, precision and subsequent F-measure score based on counting the pairs of mentions that are or are not in the same coreference chain in both system and reference. Obviously there are a large number of possible pairs from a given data set, many of which should not be linked. Empirically it has been observed that this metric tends to be lower than either the MUC-6 or B-Cubed metrics for the same data sets. As with the MUC-6 scorer, this is only defined for coreference chains of two or more mentions.

The document clustering and speaker identification communities have used metrics that may be of use within the intra-document and cross-document coreference research community. One of these metrics is called **purity** (Solomonoff et al. 1998). It is analogous to the MUC-6 and B-Cubed metrics in its adoption of a model-theoretic view, but it differs by computing the maximum of the number of mentions at the intersection of a given system/reference coreference chain alignment. A related metric is **entropy** (Dhillon et al. 2001), which measures the entropy between the mention clusters (coreference chains) of the system and the reference clusters. Neither of these scoring metrics have been used very frequently within the coreference resolution communities, but as this workshop has begun to expand the scope of coreference resolution to that of cross-document coreference and entity/document clustering (as in SPOCK and SemEval Web People tasks), we thought it appropriate to compare their utility.

6.3.2 A comparison between these metrics

The scoring methods fall into two broad categories, which we will call **set-based** methods and **entity-based** methods.

Set-based methods credit a system if it identifies part of an anahoric chain (equivalence class) and penalizes it if it missed a part. MUC scorer, B-cubed, purity and pairwise fall in this category. By contrast, entity-based methods do not credit or penalize the system for success and failure on parts of the data, but rather take a global view, evaluating how well the system succeeded in discriminating between the various entites (defined by the coreference sets). ACE score, entropy and mutual information fall in this category.

The difference between set-based metrics and entity-based methods is best illustrated with an example. Figure 6.1 shows the result of a system run: the dark circles are mentions that refer to one entity, the light circles are mentions thay refer to a second entity. The key provides anaphoric links between coreferent mentions, and the response provides a somewhat different set of links which is the system output. Looking at the correspondence between the system and the key, we notice that the system did identify some of the links correctly – for example, it identified the first four mentions as coreferent with one another – but also made a few wrong connections. This is why the set-based scoring methods give it a medium score, ranging between 0.5 and 0.75 (Table 6.3).

Entity-based measures, however, give the system a much lower score, as

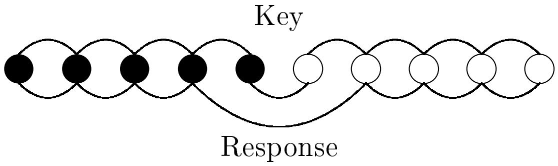


Figure 6.1: Key and response links

	Precision	Recall	F
MUC	0.750	0.750	0.750
B-cubed	0.500	0.680	0.576
Purity	0.500	0.800	0.615
Pairwise	0.414	0.600	0.490
ACE			0.125
Entropy	0.000	0.278	0.000
Mutual Information			0.000

Table 6.3: Scores given to the system in Figure 6.1

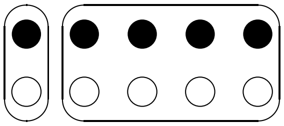


Figure 6.2: An entity representation of Figure 6.1

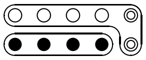
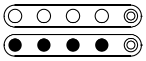
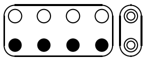
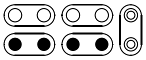
		MUC	B-cubed	Purity	Pairwise	ACE
	a.	0.933	0.846	0.889	0.765	0.583
	b.	0.800	0.775	0.847	0.727	0.567
	c.	0.933	0.750	0.750	0.619	0.542
	d.	0.833	0.750	0.750	0.556	0.167

Table 6.4: Ranking of system outputs

seen in the bottom part of Table 6.3. The reason is that while the system identified some anaphoric links correctly, it failed to distinguish between the two entities. Knowing how the system classified a particular mention gives us no information about what entity that mention refers to. This can be seen if we rearrange the mentions of Figure 6.1, and instead of drawing links we group them into sets as in Figure 6.2.

It appears that researchers in anaphora resolution prefer set-based scoring methods over entity-based methods, which are more popular in clustering. The reason is probably tht set-based methods are closer to evaluating the *process* of anaphora resolution: most systems proceed mention by mention trying to find the closest link, so it makes sense to use this approach in order to evaluate their performance.

Every measure has some pathological cases where it yields some counter-intuitive results. Table 6.4 shows the results for some specific system output which compare to a common key.

6.3.3 Implementation of the scoring metrics

We implemented a java program in which the following metrics were implemented: MUC-6, B-Cubed, Pairwise, Purity (Recall, Precision and F), and modified versions of Pairwise and MUC-6 in which singleton coreference chains could be incorporated. This scoring suite also incorporate a call-out to the ACE perl script scorer. The input was expected to be MUC-style coreference chains, so the scorer suite incorporated a utility to generate ACE-style APF versions of this data on the fly, for both system and reference data. The result was a use of the ACE Value metric in which the cost matrix was essentially completely uniform (since some of the data we were evaluating did not make ACE-appropriate distinctions of entity types nor entity mentions). The program can be called from BART.

Chapter 7

Acknowledgments

First of all, we wish to thank the US sponsors of the workshop–NSF and the Department of Defense–and the CLSP team: Fred Jelinek, Sanjeev Khudanpur, Jason Eisner, and Damianos Markatos. Without them, this would not have happened. We also wish to thank Fred, Sanjeev, Jason and Damianos for support through the workshop, for organizing a great social programme, and for general companionship; and Monique Folk, Justin Martin, and Tomas Kasparek for much invaluable help throughout with all sorts of issues.

We also wish to thank:

- EML Research, Germany for turning MMAX2 into open source and for allowing us to reuse code developed by Simone Ponzetto and Michael Strube; also, Christoph Müller for lots of help with MMAX2.

- MITRE corporation for supporting Janet Hitzeman's participation in the workshop.

- Andrew McCallum for supporting Rob Hall's participation.

- I2R Singapore, for partial support for Jian Sun and Xiaofeng Yang's participation.

- The Fondazione Bruno Kessler - IRST, and in particular Bernardo Magnini, for supporting the participation of Claudio Giuliano.

- The Department of Defense for allowing Jason Duncan to participate in our workshop.

- The Human Language Technology Center of Excellence at Johns Hopkins University, that supported the participation of Paul McNamee.

The development of the ARRAU corpus and support for Gideon Mann were provided by the EPSRC ARRAU project, UK.

Finally, we wish to thank all those people who cooperated with us through the workshop on a voluntary basis, and particularly Veronique Hoste, Sameer Pradhan, and Satoshi Sekine. Special thanks to prof. Kibrik from the Russian Academy of Sciences and his group, in particular Dr. Krasavina, who agreed to divide the effort of annotation of the RST discourse treebank.

Bibliography

C. Aone and S. W. Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 122–129, 1995a.

C. Aone and S. W. Bennett. Automatic acquisition of anaphora resolution strategies. In *Proc. AAAI Spring Symposium on Empirical Methods in Discourse*, Stanford, 1995b.

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 64–69, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-2012`.

Ron Artstein and Massimo Poesio. Identifying reference to abstract objects in dialogue. In David Schlangen and Raquel Fernández, editors, *brandial 2006: Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue*, pages 56–63, Potsdam, Germany, September 2006.

A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING/ACL*, Montreal, 1998a.

A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *Proc. of the LREC workshop on Linguistic Coreference*, pages 563–566, Granada, 1998b.

Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics,* College Park, Md., 20–26 June 1999, pages 57–64, 1999.

David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003.

M. Blume. Automatic entity disambiguation: Benefits to ner, relation extraction, link analysis, and inference. In *International Conference on Intelligence Analysis*, 2005.

Kalina Bontcheva, Marin Dimitrov, Diana Maynard, and Valentin Tablan. Shallow methods for named entity coreference resolution. *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, 2002.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47, 2006.

R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proc. EACL*, 2006.

C. Cardie and K. Wagstaff. Noun phrase coreference as clustering. In *Proc. SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pages 82–89, 1999.

D. M. Carter. *Interpreting Anaphors in Natural Language Texts.* Ellis Horwood, Chichester, UK, 1987.

Wallace L. Chafe, editor. *The Pear Stories: Cognitive, Cultural and Linguistic Aspects of Narrative Production.* Ablex, Norwood, NJ, 1980.

E. Charniak. *Towards a Model of Children's Story Comprehension.* PhD thesis, MIT, 1972. Available as MIT AI Lab TR-266.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, April–May 2000. URL `http://acl.ldc.upenn.edu/A/A00/A00-2018.pdf`.

Y. Chen and J. Martin. Towards robust unsupervised personal name disambiguation. In *Proc. of EMNLP*, 2007.

Nancy Chinchor and Beth Sundheim. Muc 6 corpus. *Message Understanding Conference (MUC) 6*, 2003.

M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, 2002.

A. Culotta, M. Wick, R. Hall, and A. McCallum. First order probabilistic models for coreference resolution. In *Proc. HLT / NAACL*, Rochester, 2007.

H. Daume and D. Marcu. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proc. HLT/EMNLP*, Vancouver, 2005.

David del Valle-Agudo, César de Pablo-Sánchez, and María Teresa Vicente-Díez. Uc3m_13: Disambiguation of person names based on the composition of simple bags of typed terms. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 362–365, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-2079`.

Pascal Denis and Jason Baldridge. A ranking approach to pronoun resolution. In *Proc. IJCAI 2007*, 2007. to appear.

I. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, G. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications.* Kluwer, 2001.

G. Doddington. Value-based evaluation of edt. Technical report of the ace 6-month meeting, Department of Defense, May 2001.

Zhou Guo Dong, Su Jian, Zhang Jie, and Zhang Min. Combining various knowledge in relation extraction. In *Proc. 43rd ACL*, pages 427–434, Ann Arbor, MI, 2005.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, Mass., 1998.

Derek Gross, James F. Allen, and David R. Traum. The Trains 91 dialogues. TRAINS Technical Note 92-1, University of Rochester Computer Science Department, July 1993. URL `ftp://ftp.cs.rochester.edu/pub/papers/ai/92.tn1.trains_91_dialogues.ps.Z`.

S. Harabagiu and D. Moldovan. Knowledge processing on extended WordNet. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 379–405. MIT Press, 1998.

W Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, pages 97–109, 1970.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 15th International Conference on Computational Linguistics,* Nantes, France, 23-28 August 1992, pages 539–545, 1992.

Peter A. Heeman and James Allen. The Trains 93 dialogues. TRAINS Technical Note 94-2, University of Rochester Computer Science Department, March 1995. URL `ftp://ftp.cs.rochester.edu/pub/papers/ai/94.tn2.trains_93_dialogues.ps.gz`.

Andrea Heyl and Günter Neumann. Dfki2: An information extraction based approach to people disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 137–140, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-2027`.

J. Hobbs. Resolving pronoun references. *Lingua*, 44:339–352, 1978a.

J. R. Hobbs. Resolving pronoun references. *Lingua*, 44:311–338, 1978b.

J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. *Artificial Intelligence Journal*, 63:69–142, 1993.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. Ontonotes: the 90% solution. In *Proc. HLT-NAACL*, 2006.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 625–632, 2006.

A. Kehler. Probabilistic coreference in information extraction. In *Proc. 2nd EMNLP*, 1997.

C. Kennedy and B. Boguraev. Anaphora for everyone: pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 113–118, Copenhagen, Denmark, 1996.

S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):525–561, 1994a.

S. Lappin and H. J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–562, 1994b.

Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In C. Fellbaum, editor, *WordNet. An Electronic Lexical Database*, chapter 11, pages 265–283. Cambridge, Mass.: MIT Press, 1998.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

X. Luo. On coreference resolution performance metrics. In *Proc. NAACL / EMNLP*, Vancouver, 2005.

X. Luo and I. Zitouni. Multi-lingual coreference resolution with syntactic features. In *Proceedings of Human Language Technology conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 660–667, 2005.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *ACL 2004*, 2004. URL `http://acl.ldc.upenn.edu/acl2004/main/pdf/243_pdf_2-col.pdf`.

Gideon Mann and David Yarowsky. Unsupervised personal name disambiguation. In *CoNLL*, 2003.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

K. Markert and M. Nissim. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3), 2005.

A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Proc. of NIPS*, 2004.

N Metropolis, M Rosenbluth, A Teller, and E Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, pages 1087–1092, 1953.

George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

Tom Minka. Estimating a dirichlet distribution. Technical report, Cambridge University, 2003.

A. Moschitti. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 335–342, 2004a.

Alessandro Moschitti. A study on convolution kernels for shallow semantic parsing. *Proceedings of the 42nd Conference on Association for Computational Linguistics (ACL 2004)*, 2004b.

Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. *Proceedings of the 17th European Conference on Machine Learning*, 2006a.

Alessandro Moschitti. Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL2006)*, 2006b.

C. Müller and M. Strube. Multi-level annotation in MMAX. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 198–207, Sapporo, July 2003. URL `http://www.sigdial.org/workshops/workshop4/proceedings/`.

Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. Technical report, Dept. of Statistics, University of Toronto., 1998.

V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111, Philadelphia, 2002a.

V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Meeting of the ACL*, 2002b.

M. Poesio. The MATE/GNOME scheme for anaphoric annotation, revisited. In *Proc. of SIGDIAL*, Boston, May 2004a.

M. Poesio and M. A. Kabadjov. A general-purpose, off the shelf anaphoric resolver. In *Proc. of LREC*, pages 653–656, Lisbon, May 2004.

M. Poesio and R. Vieira. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216, June 1998a. Also available as Research Paper CCS-RP-71, Centre for Cognitive Science, University of Edinburgh.

M. Poesio, R. Vieira, and S. Teufel. Resolving bridging references in unrestricted text. In R. Mitkov, editor, *Proc. of the ACL Workshop on Operational Factors in Robust Anaphora Resolution*, pages 1–6, Madrid, 1997. Also available as HCRC Research Paper HCRC/RP-87, University of Edinburgh.

M. Poesio, S. Schulte im Walde, and C. Brew. Lexical clustering and definite description interpretation. In *Proc. of the AAAI Spring Symposium on Learning for Discourse*, pages 82–89, Stanford, CA, March 1998. AAAI.

M. Poesio, T. Ishikawa, S. Schulte im Walde, and R. Vieira. Acquiring lexical knowledge for anaphora resolution. In *Proc. of the 3rd LREC*, Las Palmas, Canaria, 2002.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. Learning to solve bridging references. In *Proc. of ACL*, pages 143–150, Barcelona, July 2004.

Massimo Poesio. Discourse annotation and semantic annotation in the GNOME corpus. In Bonnie Webber and Donna K. Byron, editors, *Proceedings of the 2004 ACL Workshop on Discourse Annotation*, pages 72–79, Barcelona, July 2004b. Association for Computational Linguistics. URL `http://acl.ldc.upenn.edu/W/W04/W04-0210.pdf`.

Massimo Poesio and Ron Artstein. Annotating (anaphoric) ambiguity. In *Proceedings from the Corpus Linguistics Conference Series*, Birmingham, England, July 2005. URL `http://www.corpus.bham.ac.uk/PCLC/`.

Massimo Poesio and Renata Vieira. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216, June 1998b.

Simone Paolo Ponzetto and Michael Strube. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 192–199, New York City, N.Y., June 2006.

Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pages 1440–1447, Vancouver, B.C., July 2007.

Octavian Popescu and Bernardo Magnini. Irst-bp: Web people search using name entities. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 195–198, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-2041`.

Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric to semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

Herbert Rubenstein and John Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

Nuno Seco, Tony Veale, and Jer Hayes. An intrinsic information content metric for semantic similarity in WordNet. In *Proceedings of the 16th European Conference on Artificial Intelligence,* Valencia, Spain, 23–27 August 2004, pages 1089–1090, 2004.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

C. L. Sidner. *Towards a computational theory of definite anaphora comprehension in English discourse.* PhD thesis, MIT, 1979.

A. Solomonoff, A. Mielke, M. Schmidt, and H. Gish. Clustering speakers by their voices. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 757–760, 1998.

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

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

J. Steinberger, M. Poesio, M. Kabadjov, and K. Jezek. Two uses of anaphora resolution in summarization. *Information Processing and Management*, 43: 1663–1680, 2007. Special issue on Summarization.

Michael Strube and Simone Paolo Ponzetto. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1419–1424, Boston, Mass., July 2006.

Olga Uryupina. Evaluating name-matching for coreference resolution. *Proceedings of LREC'04*, 2004.

K. van Deemter and R. Kibble. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637, 2000. Squib.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

Y. Versley. Antecedent selection techniques for high-recall coreference resolution. In *Proc. of EMNLP-CONLL*, Prague, 2007.

Yannick Versley. A constraint-based approach to noun phrase coreference resolution in German newspaper text. In *Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS 2006)*, 2006.

R. Vieira and M. Poesio. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593, December 2000.

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

Daniel C. Y. Lim Wee M. Soon, Hwee T. Ng. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 21 (1):521–544, 2001.

Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics,* Las Cruces, N.M., 27–30 June 1994, pages 133–138, 1994.

X. Yang, J. Su, G. Zhou, and C. Tan. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 127–134, Barcelona, 2004a.

X. Yang, J. Su, G. Zhou, and C. L. Tan. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proc. 42nd ACL*, Barcelona, 2004b.

X. Yang, J. Su, and C. L. Tan. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL*, pages 41–48, Sydney, 2006a.

X. Yang, J. Su, and C. L. Tan. A twin-candidate model for learning based anaphora resolution. *Computational Lingustics*, To appear.

Xiaofeng Yang, Jian Su, and Chewlim Tan. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 06)*, pages 41–48, 2006b.

Stanley Wai Keong Yong and Jian Su. Effective methods of using web based information for relation extraction. In *Proc. of IJCNLP*, 2008.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(6):1083 – 1106, 2003.

S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 419–426, 2005.