UNIVERSITY OF CALIFORNIA,
IRVINE

Graphical Models for Entity Coreference Resolution

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Priya Venkateshan

Dissertation Committee:
Professor Alexander Ihler, Chair
Professor Padhraic Smyth
Professor Sharad Mehrotra

2011

DEDICATION

To my parents, for sparking the fire of my ambition and for always being there for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Master's Theses are anything but solo efforts, and this one is no exception.

I would like to firstly thank my advisor, Prof. Alex Ihler for his guidance and direction and unwavering support. I would also like to thank my committee members Prof. Sharad Mehrotra and Prof. Padhraic Smyth for their suggestions and feedback on this work.
I thank Qiang Liu, for his tremendous help with the central ideas presented in 5, and David Keator for his help in understanding and implementing Affinity Propagation.
I also thank Emili Sapena of the Departament de Llenguatges i Sistemes Informtics at Universitat Politcnica de Catalunya for his great help with feature extraction from the Semeval Dataset.

I cannot thank my mother, father and sister enough for their unwavering faith and support without which this thesis would not have been possible.

ABSTRACT OF THE THESIS

Graphical Models for Entity Coreference Resolution

By

Priya Venkateshan

Master of Science in Computer Science

University of California, Irvine, 2011

Professor Alexander Ihler, Chair

Entity Resolution, the problem of resolving token sequences in text to discourse entities, is a key problem in the natural language processing domain to statistical machine learning techniques are increasingly applied. One of the most common frameworks for viewing entity resolution as a supervised learning problem is the mention-pair model, where as a first step, a classifier is trained to make pairwise decisions for every pair of token sequences, or mentions, regarding whether they refer to the same entity or not. Then, coreference chains are formed using these pairwise decisions.

In this thesis, we present two supervised learning approaches to Entity Resolution. The first uses affinity propagation, a message-passing algorithm, to create coreference chains, taking as input pairwise probabilities of coreference given by a trained classifier. We also seek to incorporate linguistic information like part-of-speech tags. Our second approach looks at methods that enforce transitivity within coreference chains while building them, and learns parameters from maximizing a pseudo-likelihood estimate of the data conditioned over only the set of valid configurations of pairwise coreference decisions. Both our methods perform better than the tested baselines, and on par or better than the other supervised learning approaches to Entity Resolution published on the Semeval Coreference Resolution Task dataset.

Chapter 1

Introduction

Most Natural Language Processing work examines the microstructure of language, at or below the level of individual sentences. However, many tasks would benefit from information at a more global level, combining and linking different sentences or parts of the same document, or even multiple documents and document sources. One of the important subtasks central to understanding natural language at the discourse level is to find the different entities, events and other abstract notions being discussed and to identify their interconnectedness and track their references throughout. This identification of interconnectedness, or of identifying which entities and events are regarding the same discourse entities (entities being discussed in the document), which we deal with in this work, is known as **Coreference Resolution**.

The concept of Coreference Resolution can be explained using the example given in Figure 1.1. In the example, *John*, *him* and *himself*, all refer to the same entity - John, and are hence colored similarly, as are the two references to *Bob* and the three references to *the book*. The ARKRef Coreference Resolution system [1] was run on the given set of sentences. We see that it has correctly identified those noun-phrases that

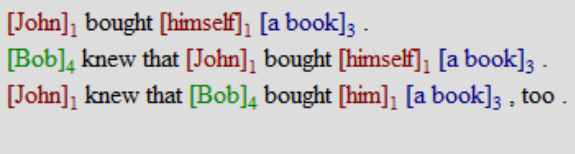


Figure 1.1: Text with entities resolved using ARKRef Coreference Resolution System [1]

refer to the same discourse entity by using, among other features, word similarity, gender and word proximity.

The real-world applications of Coreference Resolution are many. The most popular are *document summarization* and *question answering*.

Azzam, et al [7] is an example of using Coreference Resolution for document summarization. In the work, the main topic of a document, for summarization purposes, is discovered using Coreference Resolution to find the entity with the most number of references in the document.

Coreference Resolution can be used for question answering, as well. The answer to a question like *How much did Mercury spend on advertising in 1993?*, might be obtained in a sentence like *The Corporation spent USD 150,000 on advertising last year*, where *The Corporation* referred to *Mercury*, mentioned in a previous sentence. To extract this additional information, we need Coreference Resolution.

Based on whether we are resolving each reference to an entity (a person, place, organization or artifact) or to an event or an idea being discussed, there are different flavours of coreference resolution. Prominently studied among these are **Entity Coreference Resolution**, which deals with references to people, places, organizations and **Event Coreference Resolution** which tries to resolve references to events being discussed in the document. Further, Coreference Resolution which attempts to resolve entities or events that span multiple documents in a corpus is called **Cross-Document**

Coreference Resolution.

The scope of this work covers **Single-Document Entity Coreference Resolution**, where we attempt to resolve identified references - known henceforth as **mentions** - as referring to entities, which might be people, locations, organizations or artefacts.

The rest of this dissertation is organized as follows. Chapter 2 provides an overview of the task itself, the datasets we use, and the metrics we use to evaluate our approaches. In Chapter 3, we provide a brief overview of supervised approaches to Entity Resolution. Our two contributions are explained in Chapter 5, where we present a learning method that takes into account the formulation of Entity Resolution as an integer linear program, and in Chapter 4, where we present a supervised clustering approach to Entity Resolution that utilizes affinity propagation. We then summarize our contributions and results to conclude.

Chapter 2

Overview of the Entity Resolution Task

2.1 Task Definition

Entity Coreference Resolution is the task of identifying expressions of text that refer to the same discourse entity. These expressions of text that are identified as referring to any real world entity are known as **mentions**. The real-world entities that are being referred to can be Persons, Places or Organizations, and the mentions that refer to them are usually Noun-Phrases and possessive determiners.

A given document D contains a set of mentions $\mathbf{m} = m_1, m_2 \ldots m_{n-1}, m_n$, where each mention m_i specifies a span of tokens in the text of the document.

The goal of the task is to design and implement an automated system capable of assigning a discourse entity, or a topic of the current conversation which might be a person, place or organization, to each mention. Concretely, the problem can be

phrased thus: For a document D containing a set of mentions $\mathbf{m}$, an Entity Resolution system has to propose a clustering $C_1, C_2 \ldots C_k$ where each $C_i \subseteq \mathbf{m}$, and $\forall i, j$ such that $i \neq j$, we have $C_i \cap C_j = \emptyset$.

This proposed clustering $\mathbf{C} = C_1, C_2 \ldots C_k$ are known as **response entities** or **response clusters**. They are evaluated against the true clusters $\mathbf{C^*} = C_1^*, C_2^* \ldots C_{k^*}^*$, which are known as the **key entities** or **key clusters**. Here, the number of key clusters, k^* can possibly be different from the number of response clusters k.

The mentions we resolve into entity clusters can either be *gold mentions* in which case we use the same annotated mentions that occur in the key, or they can be *system mentions*, which means they are detected and extracted by an automated system [3] [4]. In case of using gold mentions, the total number of mentions in all the key and response entities are the same, as they are the same set of mentions. They however need not be the same in case of system mentions, as the sets of tokens detected by the mention-detection system can be quite different from the true mentions.

2.2 Dataset

The dataset we used was part of the Coreference Resolution in Multiple Languages Task at the 5th International Workshop on Semantic Evaluations [31], henceforth known as the Semeval Dataset. This dataset, specific to the English-Language task, was taken from the Ontonotes Release 2.0 Corpus, which contains Newswire and Broadcast News data, which have 300,000 words from the Wall Street Journal, and 200,000 words from the TDT4 collection. It is distributed by the Linguistic Data Consortium [30]. The Ontonotes corpus builds on the Penn Treebank specification for syntactic annotations, and Penn Propbank for predicate argument structures.

The semantic annotations consist of named entities, word senses (which are linked to ontology), and coreference information. The dataset is split into Training, Development and Test datasets, with the Training set consisting of 227 documents, the Development set consisting of 39, and the Test set having 85 documents.

Each document is tokenized and there are a set of features associated with each token. These features are of two types - features that are annotated by hand, and features that are predicted by an automated system. The following features are given for each word:

1. ID: Numeric identifiers for each token in the sentence
2. Token: The text of the word.
3. Lemma: A coarse identification of the token's part-of-speech.
4. Feat: These are the morphological features - features which describe morphemes, or meaningful units of language that cannot be divided further. They include part-of-speech type (noun, verb, etc), number (singlular/plural), gender(male, female, neutral), case (nominative, accusative, etc), and tense (past, present).
5. Head: This feature is the ID of the syntactic head (0 if the token is the root of the syntactic dependency tree).
6. DepRel: Dependency relations corresponding to the dependencies described in the Head column (sentence if the token is the tree root).
7. NE: Named Entity types in open-close notation. Open-close notation is when an opening paranthesis followed by the named entity type denotes the first token in a sequence of tokens making up a noun-phrase, while the named entity

type followed by a closing paranthesis denotes the last token in a sequence. Noun-phrases identified thus might embed but cannot overlap.

8. Pred: The semantic labels (person, object, organization, date, time, etc) for each predicate.

9. APreds: For each predicate in the Pred column, this feature gives its semantic roles or dependencies.

10. Coref: This column represents coreference relations in open-close notation. Every entity has an ID number. Every mention is marked with the ID of the entity it refers to. The span of mentions is denoted using open-close notation as described previously. An opening paranthesis followed by the ID number of an entity denotes the first token of a mention, while an entity ID followed by a closing paranthesis denotes the last token of a mention. Mentions can be nested and embedded within another, but they cannot cross each other. The resulting annotation is a well-formed nested structure.

Each of these features apart from *ID*, *Token*, *Coref* and *Apreds* is also additionally generated by a system. This was intended so as to be able to study the performance of system-generated features versus hand-labelled features for this task. Of these features, *Lemma* and *PoS* were generated using SVMTagger [4], while the dependency information (*Head*) and predicate semantic roles (*Pred and Apreds*) were generated with JointParser [3], which is a syntactic-semantic parser.

2.3 Evaluation Settings

The Semeval task, in order to study the effect of the usage of different levels of linguistic information, divided the task into four evaluation settings along two different

dimensions.

2.3.1 Gold Setting vs Regular Setting

In the gold setting, participants were allowed to use the manually annotated columns of the dataset, along with the gold (true) mention boundaries. In the regular setting however, participants were only allowed to use the system-generated columns. This meant mention detection had to be implemented by the participants for the regular setting. This was to determine how much high-quality preprocessing information mattered vis-a-vis machine-generated versions of the same information.

2.3.2 Closed vs Open Setting

In the closed setting, participants were only allowed to use the Semeval datasets for the task, whereas in the Open setting, the usage of external information sources like Wordnet and Wikipedia and other datasets were allowed. This was in order to see whether external information improved Entity Resolution performance.

Our systems presented here have all been evaluated according to the **Closed** and **Gold** settings.

2.4 Evaluation Measures

There are various methods used for evaluating Entity Resolution methods, and no single method is universally considerded the best. Rather, there is a proclivity to report results using multiple metrics, as each metric can provide insight that makes

up for the biases in the others.

In this dissertation, we evaluated our results on these commonly-used metrics - MUC, B-cubed and CEAF, as well as a new metric BLANC, that was introduced as part of the Semeval task. We describe each of these metrics below.

2.4.1 MUC

MUC [34] is the oldest and most widely-used measure for evaluating Coreference Resolution. It works by calculating the number of links required to be added or removed from the predicted coreference chains (response chains) to go from the response chains to the key chains. This method however fails to acknowledge singleton entities (entities with only one mention) and does not give credit for separating singletons from other entities. In an extreme case, this metric produces an invalid result when calculated on a document where all the entities are singleton entities. In yet another, if all the mentions are added to a single cluster, it gets a perfect Recall. Nonetheless, if the number of response clusters predicted is not very different from the number of key clusters, the metric is a good indication of the accuracy of the coreference resolution method used.

2.4.2 B-Cubed

The B-Cubed algorithm [8] computes Precision and Recall for each mention of the document to be evaluated. These are then combined to produce the final Precision and Recall for the document and the corpus. The precision for a mention m_i is defined as

$$Precision(m_i) = \frac{|C(m_i) \cap C^*(m_i)|}{|C(m_i)|}$$

and the recall for a mention m_i is defined as

$$Recall(m_i) = \frac{|C(m_i) \cap C^*(m_i)|}{|C^*(m_i)|}$$

where $C(m_i)$ is the coreference chain in the response that contains mention m_i, and $C^*(m_i)$ is the coreference chain in the key that contains mention m_i. As B-cubed is calculated on each mention, singletons are accounted for, unlike in MUC. However, if the dataset contains too many singletons, the scores quickly approach 100%, leaving little room for meaningful comparisons between different methods.

2.4.3 CEAF

The evaluation measure B-Cubed uses each entity more than once while aligning entities in the key and eesponse, due to which counterintuitive results can be obtained. Luo, et al [26] proposed the Constrained Entity Alignment F-Measure (CEAF) which uses a similarity function ϕ to find a one-to-one mapping between the Key and Response entities. The CEAF measure considers a similarity measure ϕ to find an alignment $g*$ between the key and response according to this similarity measure, such that

$$g* = \underset{g \in G_m}{\operatorname{argmax}} \sum_{C_i^* \in C^*} \phi(C^*, g(C^*)),$$

where G_m is the set of all one-to-one mappings between the key and response entities, and each $g(C_i^*)$ maps the entity C_i^* in the key to a corresponding entity in the response, and $\phi(a, b)$ is a measure of how similar the clusters a and b are.This similarity metric is applied on every pair of entities, one from the key, and the other from the response, to measure the goodness of each possible alignment. The best mappings are used in order to calculate the CEAF Precision, Recall and F-Measure.

Luo, et al [26]proposed the following similarity metric which is the most widely used and on which results have been widely reported. This is the similarity metric we use in our evaluation. For each pair (C_i^*, C_j) originating from the Key C^* and Response C,

$$\phi(C_i^*, C_j) = |C_i^* \cap C_j|.$$

The similarity measure is used to find the alignment with the best total similarity, denoted by $\phi(g^*)$

The Precision is specified as

$$P = \frac{\phi(g^*)}{\sum_i \phi(C_i, C_i)}$$

and the Recall as

$$R = \frac{\phi(g^*)}{\sum_i \phi(C_i^*, C_i^*)}$$

As gold mentions are used, the Precision and Recall values turn out to be the same - they both are the number of common mentions for each entity divided by the total number of mentions. However, the disadvantage of finding an alignment is that if the appropriate alignment for a response entity is not found in the list of key entities, a correct link might get ignored.

2.4.4 BLANC

The Bilateral Assessment of Noun-Phrase Coreference (BLANC) metric is based on two types of decisions taken with Entity Resolution:

1. Coreference Decisions (made by Entity Resolution system)
    (a) A Coreference link (c) holds between every two coreferent mentions

(b) A Non-Coreference link (n) holds between every two mentions that do not corefer.

2. Correctness Decisions (made by evaluation system)

   (a) A right link (r) that has the same value in both the key and eesponse (i.e. the System is correct).

   (b) A wrong link (w) that does not have the same value in both the key and response (i.e. the System is not correct).

Using these values, Precision and Recall are defined on Coreference decisions and Non-Coreference decisions:

$$\text{Coreference Precision } P_C = \frac{rc}{rc + wc}$$

$$\text{Coreference Precision } R_C = \frac{rc}{rc + wn}$$

$$\text{Non-Coreference Precision } P_N = \frac{rn}{rn + wn}$$

$$\text{Non-Coreference Recall } R_N = \frac{rn}{rn + wc}$$

The overall BLANC precision P would be the average of P_C and P_N, and the overall BLANC recall R would be the average of R_C and R_N.

$$P = \frac{P_C + P_N}{2}$$

$$R = \frac{R_C + R_N}{2}$$

We see that BLANC is similar to other metrics that evaluate the goodness of any given clustering by taking into account not only putting similar data points in the same cluster, but also putting dissimilar objects in different clusters. It does not

favour one type of decision over the other, and gives equal consideration to both coreference decisions and non-coreference decisions. Due to this, even if the response clusters of various methods contain too many singletons, the scores can range over a wide enough interval, to provide room for meaningful comparisons between the approaches, unlike B-Cubed and CEAF.

Chapter 3

Related Work on Supervised Entity Coreference Resolution

3.1 Introduction

Entity Coreference Resolution is the task of determining which sets of token in text refer to the same discourse entity. It is a widely-researched problem in the domain of natural language processing and computational linguistics. This area of research was heavily influenced by computational discourse analysis concepts like *centering* [22] and *focussing* [21], leading to the development of various *centering algorithms* in the 1970s and 1980s. The emergence of statistical techniques for natural language processing in the 1990s saw a gradual shift of Coreference Resolution approaches from heuristic-based to more learning-based approaches. Additionally, as Coreference Resolution can inherently be considered a clustering task, it generates considerable interest from the Machine Learning community as well.

Our approaches to Entity Resolution presented in this dissertation are based on su-

pervised learning approaches, and hence in this chapter, we provide a brief overview of relevant supervised learning approaches to entity resolution.

3.2 Classification of Supervised Methods in Entity Coreference Resolution

Ng [27] classifies Supervised Entity Resolution methods into three categories - *Mention-Pair Models*, *Entity-Mention Models* and *Ranking Models*. Our approaches to Entity Resolution presented in this dissertation are Mention-Pair approaches, and hence we shall give a more detailed overview of the Mention-Pair model. We also briefly outline the Entity-Mention and Ranking models in this chapter.

3.2.1 Mention-Pair Models

The Mention-Pair model is a system that, at its core, makes pairwise decisions about whether two mentions are coreferent. The model was first proposed by Aone and Bennette [5] and is widely used and researched, due to its inherent simplicity and flexibility. Despite these advantages, a model consisting of pairwise decisions has the disadvantage that it is quite possible that these independent decisions will not be consistent with each other. For example, if mentions A and B are deemed coreferent, and B and C as coreferent, there is no guarantee that the pairwise decision-making system will deem A and C as coreferent. Thus, in order to make a Mention-Pair model consistent, we need to specify a mechanism to create consistent coreference chains along with the pairwise decision-making system, where the decision-making system is usually a classifier. Another issue with the Mention-Pair model is the creation of training instances. The intuitive method to do so would be to create instances of ev-

ery pair of mentions in each document in the training dataset. However, this results in a very skewed set of training instances with negative instances constituting more than 90% of the trainingset. Some way of balancing this skew is essential. Thus, another specification for a Mention-Pair model would be its method of creating training instances. Additionally, the pairwise classification system also needs to specify the features it learns from. We delve into the work done in each of these components of a Mention-Pair model in the below sections.

Creating Training Instances

We need examples of coreferent and non-coreferent pairs of mentions in order to train a system to take pairwise decisions. This would normally mean considering every pair of mentions in each training document as an example. However, this is not the method used in practice, as it would result in a very skewed dataset. The number of positive examples would be minuscule when compared to the number of negative examples, as most pairs of mentions are non-coreferent. There are various heuristic methods of instance-creation that are used to create training instances for Entity Resolution, with the aim of reducing the skewness. Among these, Soon et al.'s [33] method is the most widely and popularly used. Given a mention m_j, this method creates a positive instance with m_j and its closest preceding coreferent mention m_i, and negative examples with m_j and every intervening mention m_{i+1},m_{i+2},... m_{j-2}, m_{j-1}. We use this method to create instances for all the approaches implemented in this work.

Training the Coreference Classifier

Once a training set is created, we can learn a coreference model using any off-the-shelf classifier. Decision trees are widely used due to their ability to handle expressive features. Memory-based learners [14] were popular, especially in early learning-based Entity Resolution systems. Maximum entropy models [15], logistic regression [17] voted perceptrons [28] and support vector machines [18] are increasingly being used because they can provide a confidence value associated with the pairwise decision. Quite a few techniques used to create coreference chains(Section 3.2.1), including graph partitioning methods and methods based on integer linear programming, utilize confidence values in ensuring consistency within the created coreference chains.

Generating Coreference Chains

After obtaining pairwise decisions or probabilities of coreference, there are many ways to create coherent coreference chains from them. The simplest methods are **Closest-First Clustering** and **Best-First Clustering**.

In Closest-first clustering, we first make a pairwise decision on every pair of mentions, regarding whether they are coreferent or not. Then, for each mention, we resolve it to the same entity cluster as the closest preceding mention (in terms of number of intermediate mentions between the two) which is coreferent with it.

In Best-first clustering, we again make a pairwise coreference decision on every pair of mentions, along with obtaining a confidence value associated with each decision. Then, for each mention, we resolve it to the same entity cluster as the preceding mention which has the highest confidence value of being coreferent with this mention. These mechanisms are flawed in that a positive pairwise decision is given more con-

sideration than a negative one by these algorithms. For example, if A and B are two mentions deemed coreferent, as are B and C, then A and C will also belong to the same entity cluster even if there is overwhelming evidence that they are not coreferent. Additionally, as negative decisions outnumber positive decisions, the coreference chains are created based on only a small subset of all the pairwise decisions.

Graph partitioning methods address this problem of consistency. Entity Resolution is cast as a graph partitioning problem with each mention in a document considered a vertex of a graph, while an edge between mentions is given a weight equal to the probability of the two mentions being coreferent. When this graph is partitioned, each partition can be considered an entity cluster.

There are various approaches using different methods to create these partitions. Bean and Riloff [10] use the Dempster-Shafer rule to score a partition based on the number of positive and negative pairwise decisions. Correlation clustering [9] is another algorithm which attempts to produce a partition that respects as many pairwise decisions as possible.

Denis and Baldridge [15] cast Entity Resolution as an Integer Linear Programming problem. Finkel and Manning [17] add transitivity constraints to this ILP. These two methods are explained in more detail in Chapter 5.

3.2.2 Entity-Mention Methods

The Entity-Mention model was introduced for a higher level of expressiveness than the Mention-Pair model. Let us take an example of an article about democratic politics. It is possible that using a Mention-Pair model, we obtain a coreference between 'Mr. Clinton' and 'Clinton', and between 'Clinton' and 'her'. However, we

know that 'Mr. Clinton' and 'her' are not coreferent. This highlights a drawback of the Mention-Pair model, which is that the later decisions are taken independent of the earlier decisions, which might result in a lack of consistency among the mentions in each coreference chain. The Entity-Mention model [12] attempts to build entity clusters as it goes through the document sequentially. At each mention, it attempts to decide whether the current mention would belong in one of the hitherto partially-formed entity cluster, or whether there is a need to create a separate cluster for this mention. This can be done by having a classifier decide if this mention corefers with each of the partially-formed entity clusters. For this, cluster-level features are used, which are more expressive than the features that are obtained from just two mentions in the Mention-Pair model. The increased expressiveness is because they describe the levels of consistency within the entity. For example, one such feature might be gender agreement, which tries to indicate whether the current mention agrees on gender with all of the mentions in the current entity, or with only some of them, or none.

3.2.3 Ranking-based Methods

The Entity-Mention model does not address the Mention-Pair model's failure to identify the most probable candidate antecedent. Thus, Ranking-based methods consider all candidate antecedents simultaneously, and follow their natural resolution strategy to resolve each anaphoric mention to the best-ranked antecedent.

The first work to deal with Ranking was Connolly et al [13], which suggested a Tournament or a Twin-Candidate model. In order to resolve a noun-phrase, every pair of candidate antecedents are considered in relation to this noun-phrase and each time, one is classified as better than the other. In the end, the noun-phrase in question is resolved to that antecedent which is classified as better the maximum number

of times. Mention rankers [16] rank all candidate antecedents simultaneously. In practice, they tend to outperform Mention-Pair models, even though they are not any more expressive, as Mention Rankers don't use cluster-level features. Rahman and Ng [29] proposed cluster ranking which takes into account cluster-level features. Here, all the preceding clusters are ranked in order of being the cluster to which the noun-phrase in question is most likely to belong to. Here, we also need to take a decision on whether a noun-phrase is anaphoric and needs to be resolved, or not. A couple of different approaches have been suggested for this. Denis and Baldrige [15] apply an anaphoricity classifier, while Rahman and Ng [29] propose a model that jointly learns to predict coreference and anaphoricity.

3.3 Participants in the Semeval-2010 Coreference Resolution Task

This section describes the other systems in the Semeval Task, that participated in the Closed Gold-Standard evaluation setting in English. Of the four systems that worked on the Closed Gold-Standard evaluation setting in English, three methods used the Mention-Pair model with Best-First clustering.

UBIU [35] is a language-independent Coreference Resolution system. It makes use of the TiMBL classifier [14], which works on the principle of memory-based learning, for pairwise classification. The pairwise decisions are converted to coreference chains using Best-First Clustering.

Sucre [24] is an end-to-end coreference resolution tool which introduced a new approach to the feature engineering of Coreference Resolution based on a relational database model and a regular feature definition language. While the system has a variety of classifiers to choose from for the Mention-Pair model, a decision-tree classifier

is found to work best. The coreference chains are created using Best-First clustering.

TANL-1 [6] is based on parse analysis and similarity clustering. It utilizes parse trees for mention detection, and a maximum-entropy classifier for classification in the Mention-Pair Model. It too utilizes best-first clustering for creation of coreference chains.

RelaxCor [32] utilizes a graphical representation of Entity Resolution, where each vertex of the graph represents a mention, and an edge between two vertices carries the weight of the probability that the corresponding mentions are coreferent. By using an algorithm called *relaxation labeling* [23], RelaxCor reduces entity resolution to a graph partition problem given a set of weighted constraints. The weighted constraint satisfaction problem is solved in order to get the final coreference partitions.

Chapter 4

Entity Resolution using Affinity Propagation

4.1 Introduction

One of the ways to view Entity Resolution is as a clustering problem. The main disadvantage of using most clustering algorithms for this particular task is the necessity to specify the number or the size of the clusters.

The clustering algorithm called Affinity Propagation [20] [19] however does not require the number or size of clusters to be explicitly specified. Affinity Propagation is a clustering algorithm which takes as input the data points as well as a series of pairwise similarities between the data points. It is an *exemplar-based* clustering algorithm like K-Medoids, which ultimately finds data points that can be considered cluster centers, around which the other points are grouped. Unlike K-Medoids clustering, however, it does not require the number of clusters to be specified. It also does not require hard decisions to be taken on cluster centers at each iteration, but propagates

soft information about the possibility of each data point being a cluster exemplar by means of message-passing, taking as input pairwise similarities between the data points. Affinity propagation essentially performs the max-sum algorithm on a factor graph model of the data in order to find an appropriate configuration of clusters and exemplars.

We formulate Entity Resolution as an affinity clustering problem. The pairwise similarities between every pair of mentions are obtained using logistic Regression. The clusters output by the affinity propagation algorithm are the coreference chains we need. Thus we can consider, our formulation to be a Mention-Pair model which uses logistic regression as its pairwise decision-making system, and affinity propagation as its chosen method of producing coreference chains.

The rest of the chapter is organized thus: we describe the technique of affinity propagation in 4.2, and go on to describe the Mention-Pair model which uses affinity propagation in 4.3. We describe the creation of training instances in 4.3.1, the features we use in 4.3.2 and the actual usage of affinity propagation in 4.3.3. We then describe the experiments we conducted and the results we obtained in 4.4.

4.2 Affinity Propagation

Let us assume that we have a set of N data points $D = \{X_1, X_2 \ldots X_N\}$ which need to be clustered, and a pairwise similarity function s where $s(i, j)$ which represents the similarity function between data points x_i and x_j. It could, for instance, be the log likelihood of x_i conditioned on x_j, or some such measure that evaluates the likelihood of x_j choosing x_i as the exemplar. The **self-similarity** $s(j, j)$, thus would be the probability that x_j chooses itself as its own exemplar. Concretely, $s(j, j)$ would be

the probability of x_j being chosen as an exemplar. Without prior knowledge, the self-similarities can be fixed to be constant values.

Let C be an $N \times N$ binary-valued matrix where the i^{th} row describes the cluster assignment of the i^{th} data point. If $C_{ik} = 1$, that means x_k is the exemplar for data point x_j. Like K-medoids clustering, Affinity Propagation seeks to maximize the sum of similarities between data points and their examplars. The similarity measures are incorporated by defining a local function

$$S_{ij}(C_{ij}) = s(i,j).C_{ij}.$$

To ensure that all configurations on C are valid, two constraint functions $\{I_i\}_{i=1}^{N}$ and $\{E_j\}_{i=1}^{N}\}$ are introduced, where

$$I_i(C_{i1} \dots C_{iN}) = \begin{cases} -\infty & \text{if } \sum_j C_{ij} \neq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$E_j(C_{1j} \dots C_N j) = \begin{cases} -\infty & \text{if } C_{jj} = 0 \text{ and } \exists i \neq j : c_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$$

I is known as the *one-of-N constraint*. I_i operates on the i^{th} row of C. It ensures that each row has only one value assigned to 1, which means that each data point is assigned only one exemplar. E enforces the exemplar consistency constraint. E_j operates on the j^{th} column of C. It ensures that if any data point has chosen x_j as its exemplar, then j too has chosen itself as its own exemplar. Thus, we attempt to find C which maximizes

$$S(C_{11} \dots C_{NN}) = \sum_{i,j} S_{i,j}(c_{ij}) + \sum_i I_i(C_{i1} \dots C_{iN}) + \sum_j E_i(C_{1j} \dots C_{Nj})$$

Affinity propagation uses a particular kind of graphical model called a **Factor Graph** to optimize S using the max-sum algorithm. Factor graphs are a way of representing the factorization of a global fuction $g(.)$ of many variables into smaller local functions. If $g(.)$ represents a function over the joint distribution over a set of random variables $x_1, x_2, \ldots x_N$, and if it factors into a set of m local functions $f_j, j = 1 \ldots m$, then

$$\log g(x_1 \ldots x_N) = \sum_{j=1}^{m} \log f_j(X_j)$$

where $X_j \subseteq \{x_1 \ldots x_N\}$ is the set of random variables over which function $f_j(.)$ is defined.

The graph for g contains a **variable node** for each variable x_i and a **factor node** for each local function $f_j(.)$. Typically, the variable nodes are represented by circular nodes, and factor nodes by square nodes. An edge connects every factor node f_j with all the variable nodes belonging to X_j.

We can derive messages for message passing algorithms, which compute the max-product and sum-product algorithms on a graph, or if we take the logarithm on g, we can extend the max-product algorithm to the max-sum algorithm. Messages are only passed between adjacent nodes in the graph. There are five messages at each variable node:, S_{ij},α_{ij}, β_{ij}, η_{ij} and ρ_{ij}. Although these messages depend on the value of C_{ij} , only the difference between the two message values can be sent, as C_{ij} is a binary variable, i.e., $C_{ij} = C_{ij}(1) - C_{ij}(0)$.

Also, since β_{ij} and ηij can be expressed in terms of the other messages, we only need send three messages:

$$S_{ij} = s(i, j)$$

$$\rho_{ij} = s(i, j) - max_{k \neq j}(s(i, k) + \alpha_{ik})$$

$$\alpha_{ij} = \begin{cases} \sum_{i \neq j} max[\rho_{ij}, 0] & \text{if } i = j \\ min[0, \rho_{jj} + \sum_{i \neq j} max[\rho_{ij}, 0]] & \text{if } i \neq j \end{cases}$$

ρ_{ij} is known as the **Responsibility Message** and α_{ij} is the **Availability message**. The Responsibility Message provides information from x_i to x_j about how suited x_i is suited to be an exemplar of x_j. x_j responds with the Availability message, which provides information about how suited x_j is to be an exemplar, given the information it has received from the other data points.

After the algorithm reaches convergence, the values of C_{ij} are found by adding all the incoming messages at C_{ij}, and setting C_{ij} to 1 if the sum is positive, and 0 otherwise.

4.3 Using Affinity Propagation for Coreference Propagation

Affinity propagation seems an ideal method for the Coreference Resolution task. Pairwise similarities for every pair of mentions can be obtained by learning a Mention-Pair model, and applying it to every pair of mentions in the test document. The advantage of this method would be that our outputs are consistent cliques which don't require further processing in order to make them coreference chains. Additionally, neither the number of clusters nor the size of the clusters need be specified in this method.

We can view our approach as a Mention-Pair model where we use Affinity Propagation to consolidate the output of our pairwise classifier. Below, we describe our Mention-Pair model.

4.3.1 Creating Training Instances

Our training instances are created by the method specified in Section 3.2.1 . We consider every mention m_j and its closest coreferent antecedent m_i as a positive instance, and every instance in between, i.e. all mentions $\{m_{i+1}, m_{i+2} \ldots m_{j-2}, m_{j-1}\}$ and m_i with m_j as a negative instance.

4.3.2 Features

A training instance consists of a set of composite features extracted from the two mentions that constitute it. These features can be specific to each mention, like whether the mentions are Pronouns, or can be a combined feature, like whether the two mention strings match. The set of features we used are the features specified in Soon et al [33].

1. **Distance**: The distance feature counts how many sentences apart the two mentions are.

2. **i/j Pronoun**: The $i - Pronoun$ and $j - Pronoun$ features are set to True or False, depending on whether the corresponding mention is a pronoun or not. This can be extracted from the 'Feat' sections of the dataset.

3. **String Match**: Articles (*a, an, the*) are removed from the mention strings, as are demonstrative pronouns like *this, that, these, those*. If the resulting strings match, the value is set to True, else, False. Thus, 'the computer' matches 'a computer', and 'the license' matches 'license'.

4. **i/j Definitive Noun-Phrase**: A definitive noun-phrase is a noun-phrase which begins with 'the'. This feature can be either true or false for a mention, de-

pending on whether the mention string begins with 'the' or not.

5. **i/j Demonstrative Noun-Phrase**: A demonstrative noun-phrase is one which begins with *this, that, these* or *those*. This feature can be either true or false.

6. **Number Agreement**: If both the mentions are singular or both are plural, this value is True, and False otherwise. The 'Feat' field of the head token of each mention is used to detect the number of each mention.

7. **Semantic Class Agreement**: The different semantic classes are organized in a hierarchy. Any noun-phrase can be of two types - 'person' and 'object'. 'Person' is divided into 'male' and 'female', while 'Object' is divided into 'organization', 'percentage', 'money', 'location', 'date' and 'time'. If there is no agreement in the hierarchy between the two mentions, this value is set to 'False', else 'True'.

8. **Gender Agreement**: The gender of a mention is detected from the 'Feat' field of the head token the mention. If both are 'male' or 'female', the value is True. If either one of the Gender values are unknown, this feature takes on the value 'Unknown'. In all other cases, it takes on the value 'False'.

9. **Both Proper Names**: If both the mentions are Proper Names, as extracted from the 'feat' field, this value is set to True, else, False.

10. **Alias**: This value is True if one of the mentions is an alias of the other, and False otherwise. There are different ways of determining this, depending on the semantic class of the mentions. In case of organizations, we first remove postmodifiers like 'Corp' and 'Ltd', and then see if one is an abbreviation of the other. In case of persons, we compare the last words in the mention phrases to see if they match. In case of dates, we attempt to extract date, month and year values from the strings and see if they match

11. **Appositive**: An appositive is a noun or noun-phrase that renames another noun-phrase beside it. To detect if one mention is in apposition to another, we use the following heuristics. When a noun-phrase is said to be appositive to another, it usually happens that both the noun-phrases are proper names, with the appositive mention not containing any verbs, and an appositive usually is separated by a comma from its closest antecedent, which it refers to. For example, in the phrase *Bill Gates, the chairman of Microsoft Corp.*, the mention *the chairman of Microsoft Corp.* is said to be appositive to *Bill Gates*.

We train a Logistic Regression classifier on the training examples created as specified above, and then apply it on every pair of mentions in each document in the Test set, in order to find the probability of two mentions being coreferent, and thus populate the Similarity matrix.

4.3.3 Tuning Self-Similarities

The other aspect of this formulation is to find methods to set the Self-Similarities for test documents. The self-similarities determine the number of entity clusters in each document. We used a few different heuristics to determine the self-similarity for each mention.

A good heuristic used for affinity propagation is to set all preferences to the same value. It is common to set this value to the average of all other similarity values.

$$s(k,k) = \frac{1}{m^2 - m} \sum_{i=0}^{m} \sum_{j=0, j \neq i}^{m} s(i,j)$$

However, applying this heuristic results in a large number of singleton clusters, as it assumes that all mentions have the same high probability of being chosen as an

exemplar. This assumption is invalid on real data, as different types of mentions differ in their chance of being chosen as an exemplar.

We know that the likelihood of a mention being considered an exemplar increases with the number of other mentions that are similar to it. This leads to our second heuristic where we can set the preference of each point to be the average similarity of all the other mentions to this mention.

$$s(k,k) = \frac{1}{m-1} \sum_{i=0, i \neq k}^{m} s(i,k)$$

This too, however, leads to a large number of clusters as the absolute probabilty of each mention being an exemplar is still high. To reduce this value, we opt to downweight all the self-similarities by a value γ. This value for γ can be obtained empirically by trying out various values on a small development dataset and choosing that value for γ which gives the best results on that dataset.

Additionally, we also can use linguistic information to set our self-similarity values. We know that pronouns have to necessarily be resolved to another mention, and cannot belong to a singleton cluster. In order to ensure that no pronoun is resolved as a singleton, we reduce the self-similarity of every pronoun to a low value, as, if a pronoun cannot be its own exemplar, it cannot be resolved as a singleton.

4.4 Experiments and Results

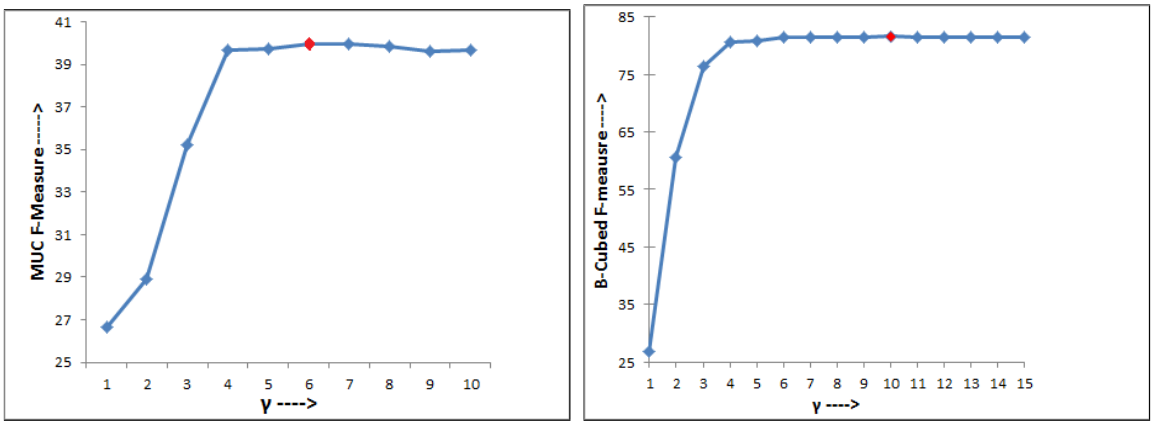
We compare our approach which performs clustering using affinity propagation against a baseline of best-first clustering. Additionally, we compare Affinity Propagation against the other systems whose results are available on the Semeval Gold-standard Closed Task, which include RelaxCor, UBIU, SUCRE, and TANL-1.

Our implementation of Best-First clustering using our set of features as described in section 3.2.1, uses Logistic Regression to obtain pairwise probabilities of coreference. We then choose that antecedent for each mention which has the highest probability among those antecedents with a coreference probability greater than 0.5.

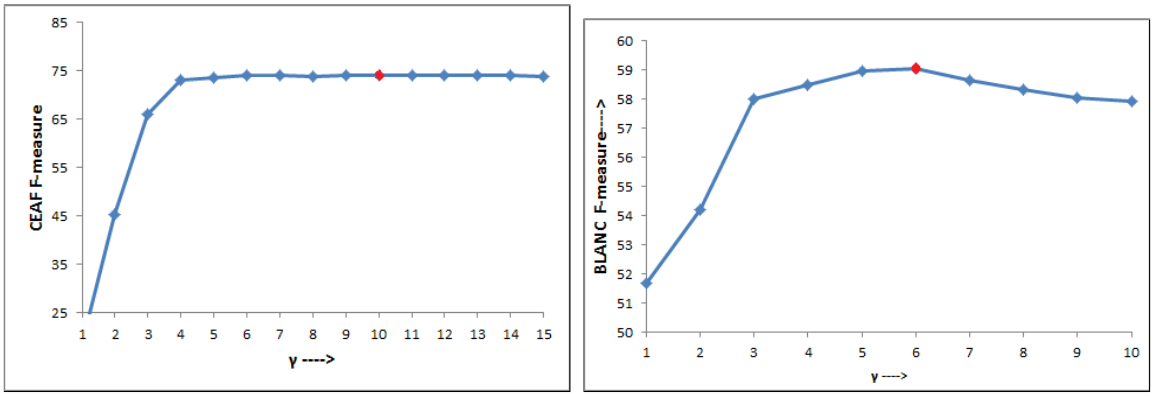
We tune the parameter γ empirically by varying it over a broad range of values while running affinity propagation on the Development dataset, which consists of 39 documents, and picking the optimal value for each of our four metrics. Thus, we find a value of γ optimal for each metric - γ_{MUC}, γ_{BCubed}, γ_{CEAF} and γ_{BLANC}. From this, we obtain four different set of results - AP_{MUC}, AP_{BCubed}, AP_{CEAF} and AP_{BLANC} each optimized for that particular metric, when we use these four values in an instance of affinity propagation.

As shown in Figure 4.1, we iterate over different values of γ in order to find the optimal value for each metric. For all metrics, we find that the F-measure of the algorithm steeply increases with increasing γ before stabilizing, as we see from the flat latter half of the curves. The optimal value of γ for all the metrics produces an F-measure only a little better than nearby γ values. The number of clusters seems to remain stable and does not increase by very much as it is limited by the reducing of pronoun self-similarities, i.e., the pronouns in the document always need to be resolved to some other mention, and thus do not add to the number of exemplars, irrespective of the value of γ. We find that γ_{MUC} and γ_{BLANC} are both equal to 6, while both γ_{CEAF} and γ_{BCUBED} are 10. This can be explained by the fact that B-Cubed and CEAF are partial to higher number of singletons, which are obtained with a higher γ value.

We present our results in Table 4.1.We compare the performances of our optimized systems against our baseline of Best-First clustering as well as against the systems described in 2.4.



(a) MUC (b) B-CUBED



(c) CEAF (d) BLANC

Figure 4.1: Varying values of F-Measures of various metrics with γ.The optimal γ value for each metric is indicated in red.

Table 4.1: Results on Affinity Propagation

X	CEAF			MUC			B Cubed			Blanc		
Exp	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Relax	75.6	75.6	**75.6**	21.9	72.4	33.7	74.8	97	**84.5**	57	83.4	61.3
Sucre	74.3	74.3	74.3	68.1	54.9	**60.8**	86.7	78.5	82.4	77.3	67	**70.8**
Tanl-1	75	61.4	67.6	23.7	24.4	24	74.6	72.1	73.4	51.8	68.8	52.1
UBIU	63.4	68.2	65.7	17.2	25.5	20.5	67.8	83.5	74.8	52.6	60.8	54
Best-First	73.15	73.15	73.15	19.69	56.97	29.27	73.65	95.65	83.22	52.57	77.5	54.38
AP_{MUC}	74.14	74.14	74.14	35.97	48.83	41.43	76.75	87.44	81.75	57.05	69.84	60.25
AP_{Bcubed}	74.18	74.18	74.18	36.76	49.18	41.41	76.65	87.66	81.79	56.88	69.78	60.06
AP_{CEAF}	74.18	74.18	74.18	36.76	49.18	41.41	76.65	87.66	81.79	56.88	69.78	60.06
AP_{BLANC}	74.14	74.14	74.14	35.97	48.83	41.43	76.75	87.44	81.75	57.05	69.84	60.25

We compare the various approaches on the basis of the F-Measure for each metric. A higher F-measure is considered better across metrics. CEAF and B-Cubed are partial to methods that perform well on detecting singletons, whereas MUC is partial to those that detect coreferent links correctly. BLANC averages both coreference and non-coreference decisions and does not lean towards either.

Best-first clustering seems better than our optimized methods in detecting singletons, as seen by the marginally higher B-Cubed scores, but our scores on the MUC metric are much higher across all our optimized methods, showing that Affinity Propagation is better than best-first clustering in identifying coreference links. When both coreference and non-coreference decisions are given equal weight during evaluation, as in the BLANC scores, we see our overall performance evaluates better than Best-first clustering, on the Semeval dataset.

We further observe that Affinity Propagation does consistently better than TANL-1 and UBIU on all four metrics. While we outperform RelaxCor in identifying coreference links as evidenced by our higher MUC scores, RelaxCor performs better on identifying non-coreferencce links, and thus has higher B-Cubed and CEAF scores than Affinity Propagation. Overall, on the Semeval Dataset, RelaxCor outperforms Affinity Propagation on the BLANC metric.

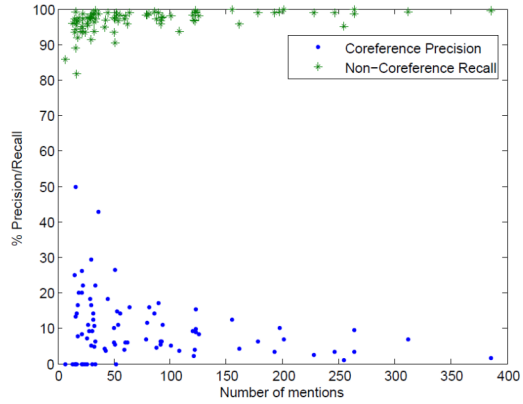
SUCRE is marginally better than Affinity Propagation on identifying non-coreference

links, while doing quite significantly better than our and all other methods we compare against on identifying coreference links. Thus, while our results are similar to that of RelaxCor, SUCRE significantly outperforms affinity propagation. In order to further understand the types of errors each algorithm makes, we analyze the effect of the number of mentions in a document versus the percentage of errors each algorithm makes in identifying coreference and non-coreference links. We choose to compare affinity propagation with best-first clustering as well as with RelaxCor and Sucre and observe what factors might be contributing to the difference in performance between these algorithms and affinity propagation.

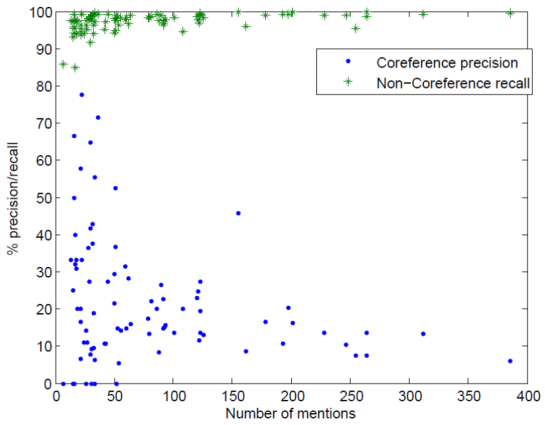
We plot graphs of the attraction precision and repulsion precision for each document in the test dataset versus the number of mentions to be resolved in that particular document. Attraction Precision is the ratio of the number of pairs of mentions correctly deemed coreferent by the algorithm to the number of coreferent pairs of mentions as given by the set of key mentions. Simlarly, Repulsion Precision is the ratio of the number of pairs of mentions correctly deemed non-coreferent by the algorithm to the number of non-coreferent pairs of mentions as are given by the set of key mentions. These are represented as percentages in the graph rather than as a value between 0 and 1. The higher the values of these figures for each document, the better its performance is deemed to be.

Our first comparison is between affinity propagation and our baseline, best-first clustering. We can see from a cursory glance of the graphs that affinity propagation performs better than best-first clustering on documents of all sizes. However, we can also see that the difference in performance decreases with increase in the number of mentions in the document. There is little or no difference in non-coreference precision performance between the two algorithms.

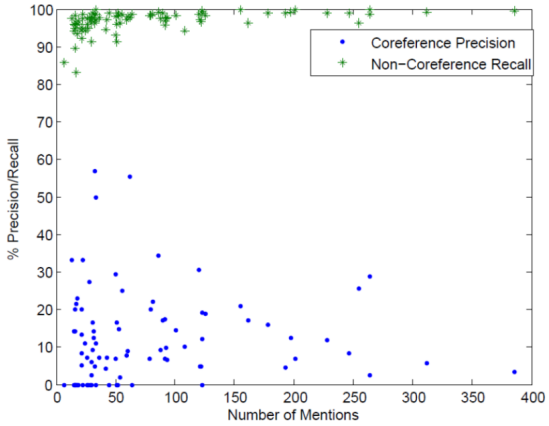
We can also see that affinity propagation's performance on detecting coreference links



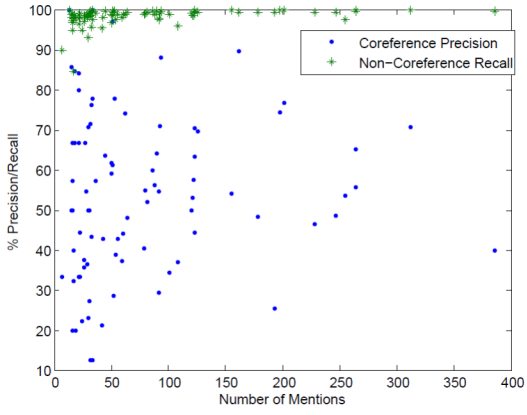
(a) Best-First Clustering



(b) Affinity Propagation



(c) RelaxCor



(d) SUCRE

Figure 4.2: Plots of coreference and non-coreference precision for documents in the test set for each algorithm

are consistently better than that of RelaxCor, on an average. There happen to be several smaller documents where our algorithms do not get even a single coreference link right. From the graph, we see that affinity propagation produces fewer such documents when compared to RelaxCor. Affinity propagation shows slightly better performance in detecting non-coreference links for a few really small documents, but for most documents, the performance of different algorithms is similar when it comes to detecting non-coreferent links.

Thus, even though RelaxCor and affinity propagation show similar scores in all metrics other than MUC, we see from the graph that this is in large part due to the high weight the metrics other than MUC place on non-coreference links. On smaller documents, affinity propagation is clearly the better algorithm when compared to RelaxCor, though both algorithms perform similarly on average as the number of mentions in a document increases, though performance varies for different documents.

When we compare affinity propagation with SUCRE, we see that SUCRE is the better algorithm, as it has comparable values for non-coreference link precision and much higher values for coreference precision on documents of all sizes. It is observed that unlike other documents, SUCRE manages to obtain coreference precisions higher than 0 for all documents. It is also observed that while SUCRE has high values of coreference precision for documents of less than 50 mentions, it only gets higher on an average as the number of mentions increases, unlike the other algorithms where the coreference precision decreases on average as the number of mentions increases. Thus, for tasks involving smaller sized documents, like entity resolution on web snippets, an algorithm like affinity propagation might be a good choice, we need SUCRE for tasks like document summarization, which involve large documents.

4.5 Conclusions

We have presented a novel formulation of Entity Resolution as a clustering problem solved using Affinity Propagation. We also incorporate linguistic knowledge into setting the self-similarities for each mention, by setting that for each pronoun to a low value. We achieve reasonably good results which are comparable to that of the other participants of the Semeval Coreference Resolution Task. Our performance on detecting singletons needs to be improved, as seen from the CEAF and B-Cubed results. We also perform quite well on small documents, whereas there is room for improvement on larger documents with more than 100 mentions.

Further directions for this approach include incorporating Anaphora Resolution to determine self-similarities, as well as experimenting with a more expressive feature set like Sucre [24] and classification algorithm in order to obtain improved pairwise potentials.

Chapter 5

Constraint-aware Logistic Regression for Coreference Resolution

5.1 Introduction

Pairwise methods, as we have seen in Chapter 3, are the most common and best-researched approaches to Entity Coreference Resolution, where decisions are taken about whether each pair of mentions corefer or not, following which a method to derive coreference chains from those pairwise decisions is applied. Often, we see that learning from data does not affect the Coreference Resolution process beyond the making of pairwise decisions, i.e., the method to extract coreference chains improves with better data to learn from only if it improves the pairwise decisions. It would be a possible improvement if the learning more directly affected the creation of coreference chains, for example, if the learning method took the method of resolving coreference

chains into consideration.

Denis and Baldridge [15] had proposed a pairwise method where they used a maximum entropy classifer for obtaining the pairwise probabilities of coreference. This probability was then considered the edge-weight in the graph containing all mentions as nodes. The creation of coreference chains is then cast as an optimization problem and Integer Linear Programming (ILP) is used to find the coreferent edges, and hence chains in the graph.

Finkel and Manning [17] improved on this approach adding transitivity constraints to the ILP, thus enforcing consistency in the coreference chains.

In this chapter, we propose a change to the objective function of logistic regression used by Finkel and Manning [17] to take into account the fact that we would like the parameters we are learning to be learned conditioned over only the valid configurations instead of all possible configurations of the pairwise labels. We use the notion of pseudo-likelihood to account for learning the probability of each pairwise decision given the rest of the graph.

The rest of the chapter is organized thus: Section 5.2 gives an overview of the method proposed by Denis and Baldridge and Finkel and Mannings extension to the idea. Section 5.3 is where we propose our learning method which we call Constraint-aware Learning. We describe our baselines and experiments in section and discuss our results in Section 5.4

5.2 Enforcing Transitivity on Coreference Chains

Denis and Baldridge [15] cast Entity Resolution as an Integer Linear Programming problem. Each mentions m_i is considered a node in a graph, and an edge exists between every pair of mentions(nodes), with the edge weight of the edge being the probability of the two mentions it connects being coreferent. These pairwise probabilities can be obtained using any classifier that provides confidence values for its decisions. When two mentions m_i and m_j are deemed coreferent, the weight of the edge between these nodes in the graph, y_{ij} is set to 1, and if not, is set to 0. These decisions are taken such that they optimize a given objective function, subject to some constraints. In this case, the objective function is the log likelihood of the data.

The basic formulation of Entity Resolution as an ILP as given by Denis and Baldridge simply attempts to maximize the log probability of the data, with no added constraints. Let our classifier produce probabilities $P(y_{ij} = 1|m_i, m_j)$. The cost of committing to a coreference link would be

$$w_{ij} = -\log(P(y_{ij} = 1|m_i, m_j)).$$

The cost of choosing not to establish a coreference link would be

$$\bar{w}_i = -\log(1 - P(y_{ij} = 1|m_i, m_j)).$$

Thus, the objective function we are trying to minimize here would be

$$\min \sum_{<i,j>\in \mathbf{m}\times\mathbf{m}} w_{ij}.y_{ij} + \bar{w}_{ij}.(1 - y_{ij})$$

subject to

$$y_{ij} \in \{0, 1\} \forall ij \in \mathbf{m} \times \mathbf{m}$$

where y_{ij} are indicator variables set to 1 if m_i and m_j are coreferent, and 0 if not. This objective function is minimized rather than maximized as model probabilities are transformed by $-\log$.

Using this objective function, the ILP merely tries to find a global assignment which maximally agrees with the decisions made by the coreference classifier. Concretely, since the link predictions are not correlated, this simply amounts to predicting that those links exist (i.e. only those pairs of mentions corefer) for which the pairwise classifier returns a probability greater than 0.5.

One of the widely cited disadvantages of the Mention-Pair model is that there is little to ensure consistency within the Response coreference clusters. For example, in a document, *Bill Clinton* might be deemed coreferent with *Clinton*, and *Clinton* might be deemed coreferent with *Hillary Clinton*. It is quite possible that all three of these mentions might be added to the same coreference chain irrespective of the pairwise decision between *Bill Clinton* and *Hillary Clinton*.

One of the ways to ensure consistency is to require the pairwise decisions to be transitive: if mention m_i is coreferent with m_j and mentions m_j and m_k are coreferent, it should necessarily mean that mentions m_i and m_k are coreferent.

Finkel and Mannning [17] use logistic regression as their classifier of choice for the pairwise decisions to be taken. They add transitivity constraints to the previously formulated ILP. The additional constraints are:

$$\forall i, j, k \in \mathbf{m} \times \mathbf{m} \times \mathbf{m}, (1 - y_{ij}) + (1 - y_{jk}) \geq (1 - y_{ik})$$

which ensures that the graph gets partitioned into cliques, where every mention is connected to every other mention by an edge of weight 1, and each clique can be

considered a response cluster.

5.3 Constraint-aware Logistic Regression

Suppose that we have variables y_{ij} that represent each co-reference, so that

$$y_{ij} = \begin{cases} 1 & \text{if } i, j \text{ are co-referent} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

We use a vector of features X_{ij} for each (i, j) to predict y_{ij}. As discussed previously, the values of X_{ij} may depend on the characteristics of mention i, mention j, or the two together (such as gender or plurality agreement).

If the $\{y_{ij}\}$ are assumed to be independent, the model can be trained using a maximum likelihood approach, where the log-likelihood is given by

$$\mathrm{LL}(y\,;\,\theta) = \sum_{ij} \log p_{ij}(y_{ij}) \qquad\qquad p_{ij}(y_{ij}) = \frac{\exp(\theta^T X_{ij} y_{ij})}{1 + \exp(\theta^T X_{ij})} \tag{5.2}$$

Note that $y_{ij} \in \{0, 1\}$, and so p_{ij} is the usual logistic likelihood of y_{ij}.

However, since we plan to enforce consistency among the individual decisions y_{ij}, this is not the log-likelihood of the model we plan to use in practice. The "correct" model can be defined using a function $C(\cdot)$ over all y jointly to enforce consistency:

$$C(y) = \begin{cases} 1 & \forall i, j, k \quad y_{ik} = y_{jk} = 1 \;\Rightarrow\; y_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

The log-likelihood of this model is then given by

$$LL(y\ ;\ \theta) = \sum_{ij} \log p_{ij}(y_{ij}) - \log \sum_{y'} C(y') \prod_{ij} p_{ij}(y'_{ij}) \tag{5.4}$$

Note that the observed training data y satisfies consistency automatically, but that consistency changes the normalization constant of the distribution, making the log-likelihood a complex and difficult to optimize joint function over all y.

To simplify the optimization, we use a *pseudo-likelihood* approach [11]. More precisely, we define a composite likelihood estimate [25] (an extension of pseudo-likelihood), defined over the sets of edges connected to each mention i. The composite likelihood is given by

$$CL(y\ ;\ \theta) = \sum_{i} \log p(y_{i*} \mid y_{\neg i}) \tag{5.5}$$

where $y_{i*} = \{y_{i1}, \ldots, y_{in}\}$ is the set of edges connected to mention i, and $y_{\neg i}$ are those edges that are not incident to mention i, $y_{\neg i} = y \setminus y_{i*}$.

We can then see that

$$p(y_{i*} \mid y_{\neg i}) = \frac{\prod_j p_{ij}(y_{ij})}{\sum_{y'_{i*}} C(y_{i*}|y_{\neg i}) \prod_j p_{ij}(y_{ij})} \tag{5.6}$$

Note that again, the fact that our observed data satisfies consistency is used in the numerator. The denominator is simply the sum over all *consistent* configurations y'. Denoting this set as C_i, i.e., $y'_{i*} \in C_i$ if for all i, j, k, we have that if $y_{jk} = 1$ then $y_{ij} = y_{ik}$ (either 0 or 1), and if $y_{jk} = 0$ then at most one of y_{ij} and y_{ik} are 1. The composite likelihood is then

$$CL(y\ ;\ \theta) = \sum_{i} \log \frac{\exp \sum_j \theta^T X_{ij} y_{ij}}{\sum_{y'_{i*} \in C_i} \exp \sum_j \theta^T X_{ij} y'_{ij}} \tag{5.7}$$

and the composite likelihood estimator can be found by maximizing $CL(y\ ;\ \theta)$ over θ. Moreover, CL is a convex function of θ and we optimize it using a Newton-Raphson update.

5.4 Experiments and Results

We evaluated Constraint-Aware Logistic Regression-based ILP against two baselines. The first one was Denis and Baldridge's basic ILP formulation, which basically involved marking pairs of mentions as coreferent if the probability of their coreference as predicted by the pairwise Logistic Regression classifier exceeded 0.5. The second one was Finkel and Manning's ILP formulation with transitivity constraints included, for which we used Logistic Regression for the pairwise decisions, and Gurobi Optimizer [2] to solve the integer linear program. Constraint-Aware Logistic Regression was implemented using the L-BFGS optimizer from SciPy's optimization module.

We present our results below, with Denis and Baldridge's baseline referred to as *DnB*, Finkel and Manning's baseline referred to as *FnM*, and our approach as *C-Log*. We also compare our approaches to the other systems we had previously described in Section 2.4

Table 5.1: Comparison of Constraint-Aware Logistic Regression with other baselines

X	CEAF			MUC			B Cubed			Blanc		
Exp	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Relax	75.6	75.6	**75.6**	21.9	72.4	33.7	74.8	97	**84.5**	57	83.4	61.3
Sucre	74.3	74.3	74.3	68.1	54.9	**60.8**	86.7	78.5	82.4	77.3	67	**70.8**
Tanl-1	75	61.4	67.6	23.7	24.4	24	74.6	72.1	73.4	51.8	68.8	52.1
UBIU	63.4	68.2	65.7	17.2	25.5	20.5	67.8	83.5	74.8	52.6	60.8	54
DnB	73.9	73.9	73.9	30.58	61.01	40.74	76.75	87.07	81.59	55.56	77.45	59
FnM	73.45	73.45	73.45	35.51	46.03	40.09	76.84	86.12	81.26	56.74	67.94	59.66
C-log	74.23	74.23	74.23	35.76	47.74	40.97	75.17	93.76	83.44	56.78	68.88	59.84

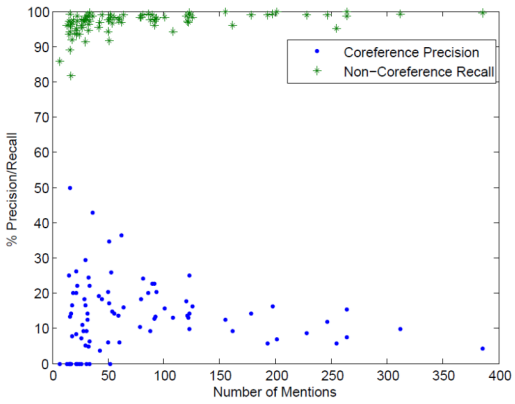
Method	ILP solve time
Denis and Baldridge	0.64
Finkel and Manning	47
C-log	38

Table 5.2: Comparison of ILP-solve time (in seconds) between the different ILP-based methods, over the test set of 85 documents

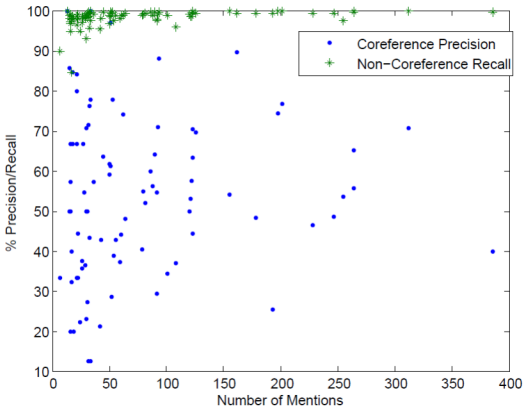
Our method C-Log does consistently better than our baselines Denis and Baldridge and Finkel and Manning on the F-measures of all the four metrics. While it outperforms the baselines on these metrics marginally, we observed that the ILPs generated by our method took lesser time to solve as compared to Finkel and Manning, as we show in Table 5.2. This, we speculate, can be attributed to our method's ability to accept a relatively small set of strong predictions in lieu of a large number of correct predictions, which results in an ILP that is simpler to solve.

We also compare our performance with that of other approaches published on the Semeval datset. We perform consistently better than TAN-L1 and UBIU on all four metrics. However, SUCRE performs better than us on all metrics except B-cubed. From its high MUC scores, we can infer that it identifies coreference links better than our method, while performing similarly on non-coreference links and singletons, as seen by our CEAF and B-Cubed scores being similar. Our performance in identifying coreference links is better than that of RelaxCor, as evidenced by our higher MUC scores, but RelaxCor performs a little better on identifying non-coreference links and singletons, as seen by the slightly higher B-Cubed and CEAF scores. Overall, RelaxCor performs a little better than us, as can be seen by our similar BLANC scores, where both types of decisions are given equal weightage during evaluation.

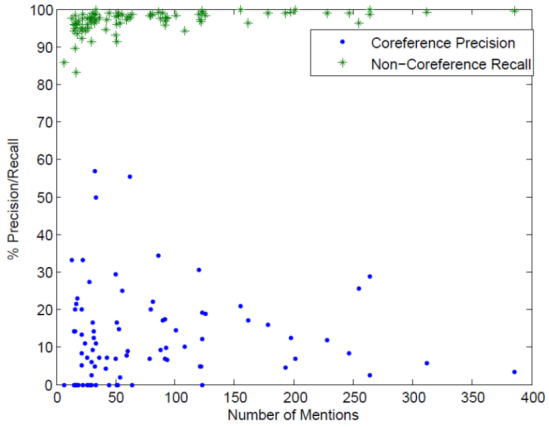
In order to further understand the types of errors each algorithm makes, we analyze the effect of the number of mentions in a document versus the percentage of errors each algorithm makes in identifying coreference and non-coreference links. We choose



(a) C-Log



(b) SUCRE



(c) RelaxCor

Figure 5.1: Plots of coreference and non-coreference precision for documents in the test set for each algorithm

to compare C-Log with AffProp as well as with RelaxCor and SUCRE and observe what factors might be contributing to the difference in performance between these algorithms and C-Log. We plot graphs of the attraction precision and repulsion presicion for each document in the test dataset versus the number of mentions to be resolved in that particular document. Attraction Precision is the ratio of the number of pairs of mentions correctly deemed coreferent by the algorithm to the number of coreferent pairs of mentions as given by the set of key mentions. Simlarly, Repulsion Precision is the ratio of the number of pairs of mentions correctly deemed non-coreferent by the algorithm to the number of non-coreferent pairs of mentions as are given by the set of key mentions.

These are represented as percentages in the graph rather than as a value between 0 and 1. The higher the values of these figures for each document, the better its performance is deemed to be.

Our first comparison is between C-Log and RelaxCor. There is very little margin to compare the two algorithms on non-coreference precision, and we can say that both algorithms perform comparably on that metric except for a few smaller documents where RelaxCor seems to perform better. When we compare on the basis of coreference precision, we see that the value of this metric is zero for fewer documents when C-Log is used, than when RelaxCor is used. But we also see that C-Log has fewer documents on which the performance is better than average, though C-Log is marginally better on small-sized documents. C-Log shows more consistent performances on documents than RelaxCor does. Thus, the consistent performance of C-Log on coreference precision is the reason C-Log is better on MUC than RelaxCor is, while the better performance of RelaxCor on smaller documents is why it is the better algorithm on the B-Cubed and CEAF metrics.

We then compare C-Log to SUCRE and note the differences in the performances. When we compare C-Log with SUCRE, we see that SUCRE is the better algorithm, as it has comparable values for non-coreference link precision and much higher values for coreference precision on documents of all sizes. It is observed that unlike other algorithms, SUCRE manages to obtain coreference precisions higher than 0 for all documents. It is also observed that while SUCRE has high values of coreference precision for documents of less than 50 mentions, it only gets higher on an average as the number of mentions increases, unlike the other algorithms where the coreference precision decreases on average as the number of mentions increases.

5.5 Conclusions

We presented a modified likelihood function to learn a classifier, which took into account the transitivity constraints that would later be imposed on the decisions made by the classifier. This was done by conditioning the pseudo-likelihood on only valid configurations of the training labels. The notion of pseudolikelihood was utilized for this. We demonstrated that our approach performs better than our baselines, as well as comparably with current systems, and that the integer linear program built using our probabilities took significantly less time to solve.

We can say that our method shows promising results compared to the current state-of-the-art in supervised approaches for entity resolution, and has scope for significant improvement in the future.

Chapter 6

Conclusion

This thesis proposed two different graph-based approaches to tackle the problem of Entity Resolution. Both approaches outperformed baselines on various metrics and perform comparably to other approaches with results reported on the same datasets. This chapter summarizes our contributions and suggests future directions for the ideas presented in this thesis.

6.1 Summary Of Contributions

We utilized the concept of Pseudo-likelihood in order to learn a set of parameters for Logistic Regression that would keep in mind the method of generating consistent coreference chains. This method improves on previous approaches that utilize Integer Linear Programming. It is also a good beginning to explore among approaches that take into consideration the method of generating coreference chains while learning a classifier.

Our other contribution was the formulation of Entity Resolution as a clustering prob-

lem using Affinity Propagation. This solves the problem of ensuring consistency among the mentions belonging to a culture, as well as the problem faced by most clustering approaches about setting the number or size of clusters. We also demonstrated the use of linguistic information, like Part-Of-Speech tags, in setting self-similarities for each mention. We perform significantly better than some of the existing approaches.

6.2 Future Work

In the near future, it would be interesting to observe the performance of these approaches on larger, standardized datasets like the MUC and ACE corpora. Additionally, a good direction to explore would be to learn and predict the self-similarities for Affinity Propagation. It might be a significant step to use Anaphora Resolution to do so, as mentions that are deemed anaphorae can have their self-similarity set to an appropriately low value, due to the fact that anaphorae cannot be singleton clusters and need to be resolved to some antecedent.

Another possible direction to explore for both our approaches would be to experiment with adding more linguistic constraints, like number or gender, to see if they significantly improve performance. This is a logical next step, as both our approaches are inherently flexible with respect to addition of more constraints to the process of creating coreference chains.

Bibliography

[1] Arkref noun-phrase coreference system, http://www.ark.cs.cmu.edu/arkref/.

[2] Gurobi optimizer, http://www.gurobi.com.

[3] Jointparser, http://nlp.lsi.upc.edu/jointparser/demo/.

[4] Syntactic-semantic parser, http://www.lsi.upc.edu/nlp/svmtool.

[5] C. Aone and S. W. Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 122–129, 1995.

[6] G. Attardi, S. Dei Rossi, and M. Simi. Tanl-1 : Coreference resolution by parse analysis and similarity clustering. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 108–111. Association for Computational Linguistics, 2010.

[7] S. Azzam, K. Humphreys, and R. Gaizauskas. Using coreference chains for text summarization. In *ACL Workshop on Coreference and its Applications*, 1999.

[8] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *Message Understanding Conference*, 1998.

[9] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[10] D. Bean and E. Riloff. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proc. Human Language Technology Conf. / North American Chapter of the Assoc. for Comput. Linguistics Annual Meeting (HLT-NAACL04)*, pages 297–304, 2004.

[11] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

[12] C. Cardie and K. Wagstaff. Noun phrase coreference as clustering. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89. Cornell University, 1995.

[13] D. Connolly, J. D. Burger, and D. S. Day. A machine learning approach to anaphoric reference. *New Methods in Language Processing*, pages 133–143, 1997.

[14] W. Daelemans and A. Van Den Bosch. Memory-based language processing. In *Computational Linguistics*, volume 11, pages 559–561. Cambridge University Press, March 2005.

[15] P. Denis and J. Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243, Rochester, New York, April 2007. Association for Computational Linguistics.

[16] P. Denis and J. Baldridge. A ranking approach to pronoun resolution. In *Linguistics*, pages 1588–1593, 2007.

[17] J. R. Finkel and C. D. Manning. Enforcing transitivity in coreference resolution. In *ACL (Short Papers)*, pages 45–48. The Association for Computer Linguistics, 2008.

[18] T. Finley and T. Joachims. Supervised clustering with support vector machines. *Proceedings of the 22nd international conference on Machine learning ICML 05*, pages 217–224, Aug. 2005.

[19] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[20] I. E. Givoni and B. J. Frey. A binary variable model for affinity propagation. *Neural Computation*, 21:1589–1600, June 2009.

[21] B. Grosz. The representation and use of focus in a system for understanding dialogues. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence Cambridge Massachusetts*, pages 67–76. William Kaufmann, Inc., 1977.

[22] B. J. Grosz, A. K. Joshi, and S. Weinstein. Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting*, pages 44–50. Association for Computational Linguistics, 1983.

[23] R. A. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:267–287, March 1983.

[24] H. Kobdani and H. Schutze. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 92–95. Association for Computational Linguistics, 2010.

[25] B. G. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80:221–239, January 1988.

[26] X. Luo. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[27] V. Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the ACL*, pages 1396–1411, 2010.

[28] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and inference over constrained output. In *Proceedings of IJCAI 2005*, pages 1124–1129, 2005.

[29] A. Rahman and V. Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977. Association for Computational Linguistics, Aug. 2009.

[30] M. Recasens, L. Marquez, E. Sapena, M. A. Marti, and M. Taule. Semeval-2010 task-1 ontonotes english: Coreference resolution in multiple languages, http://www.ldc.upenn.edu/catalog/catalogentry.jsp?catalogid=ldc2011t01.

[31] M. Recasens, L. Marquez, E. Sapena, M. A. Marti, M. Taule, V. Hoste, M. Poesio, and Y. Versley. Semeval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 1–8, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[32] E. Sapena, L. Padr, and J. Turmo. Relaxcor: A global relaxation labeling approach to coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 88–91. Association for Computational Linguistics, 2010.

[33] W. M. Soon, H. T. Ng, and C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27:521–544, April 2001.

[34] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[35] D. Zhekova and S. Kubler. Ubiu: A language-independent system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 96–99. Association for Computational Linguistics, 2010.