

A Modular Framework for Coreference Resolution

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Hamidreza Kobdani

aus Mechhed/Iran

Hauptberichter: **Prof. Dr. Hinrich Schütze**

Mitberichter: **Prof. Dr. Massimo Poesio**

Tag der mündlichen Prüfung: 10. Feb 2012

Institut für Maschinelle Sprachverarbeitung der Universität Stuttgart

2012

Abstract

Coreference resolution is playing an increasingly important role in a wide range of disciplines such as theoretical, corpus and computational linguistics. It has been shown that it is beneficial in a number of natural language processing applications, including machine translation, automatic abstracting, information extraction and question answering. As a result, it has enjoyed increased interest in recent years.

First, this thesis introduces a modular supervised system for coreference resolution. It is composed of separate, interchangeable components, between which there are clear well-defined logical boundaries that improve maintainability. This system has been used successfully in two international shared tasks on coreference resolution achieving good performance. The good performance of our system demonstrates the general validity of our design.

In addition, a new framework for feature engineering of natural language processing will be presented that is based on a relational data model of text. It includes fast and flexible methods for implementing and extracting new features, thereby reducing the effort of creating an NLP system for a particular task. This thesis presents an instantiation and evaluation of the framework for the problem of coreference resolution in multiple languages. Competitive results were able to be obtained in a short implementation period. This demonstrates the potential power of this framework for feature engineering.

An unsupervised framework will also be presented that bootstraps a complete coreference resolution system from *word associations* mined from a large unlabeled corpus. I will show that word associations are useful for coreference resolution – e.g., the strong association between *Obama* and *President* is an indicator of likely coreference. Association information has so far not been used in coreference resolution because it is sparse and difficult to learn from small labeled corpora. Since unlabeled text is readily available, the unsupervised approach proposed here addresses the sparseness problem. In a self-training framework, I train a decision tree on a corpus that is automatically labeled using word associations. I will show that this unsupervised system has better coreference resolution performance than other learning approaches that do not use manually labeled data.

Zusammenfassung

In den letzten Jahren hat Koreferenzresolution eine zunehmend wichtige Rolle in verschiedenen Disziplinen gespielt. Es hat sich gezeigt, dass die Anwendung der Koreferenzresolution auf viele Aufgaben wie z.B. der maschinellen Übersetzung, dem automatischen Zusammenfassen und der Informationsextraktion vorteilhaft ist.

In dieser Dissertation stellen wir zuerst ein modulares System für überwachte Koreferenzresolution vor. Es besteht aus verschiedenen, austauschbaren Komponenten, zwischen denen es klar definierte logische Grenzen gibt, die die Wartbarkeit verbessern.

Dieses System erzielte bei zwei internationalen Koreferenzresolutionswettbewerben gute Ergebnisse, was die Vorteile unseres Designs zeigt.

Außerdem präsentieren wir, ein neues Framework für die Merkmalsverarbeitung natürlicher Sprache, das auf einem relationalen Datenmodell basiert. Es umfasst eine schnelle und flexible Methode für die Umsetzung und das Extrahieren von neuen Funktionen und verringert dadurch den Aufwand der Erstellung eines konkreten NLP-Systems. Die vorliegende Arbeit stellt eine Instanziierung und Auswertung des Frameworks für das Problem der Koreferenzresolution in mehreren Sprachen dar. Wir konnten in einer kurzen Bearbeitungszeit gute Ergebnisse erzielen, was das Potenzial unseres Frameworks für die Merkmalsverarbeitung unterstreicht.

Wir präsentieren auch ein unüberwachtes System, in dem ein komplettes Koreferenzresolution-System aus den Wort-Assoziationen, die aus einem großen unannotierten Korpus extrahiert worden sind, gebootstrapt wird. Wir zeigen, dass Wort-Assoziationen erfolgreich in Koreferenzresolution eingesetzt werden können. So ist die starke Assoziation zwischen *Merkel* und *Bundeskanzlerin* ein Indikator für eine höhere Koreferenzwahrscheinlichkeit.

Die Anwendung von Koreferenzresolution im großen Stil scheiterte bisher am Fehlen ausreichend großer annotierter Datensätze. Für unseren unüberwachten Ansatz benötigen wir nur unannotierten Text, der für die meisten Sprachen in ausreichender Menge zur Verfügung steht.

Mit der Assoziationsinformation können wir einen Textkorpus automatisch annotieren und dann iterativ unser überwachtes System darauf trainieren und

den Korpus neu annotieren. Wir zeigen, dass das resultierende unüberwachte System besser als alle uns bekannten unüberwachten Lernmethoden funktioniert.

Acknowledgments

There are many people whom I am deeply indebted to, and without whom this work would never have been finished.

First and foremost, I would like to thank my adviser Hinrich Schütze, for his constant support. Hinrich has been more than an adviser, he has been a teacher, a mentor, a guide, an encourager, a friend and one of the best co-authors a person could have.

It has been a privilege to have Massimo Poesio as my external examiner, and I would like to thank him for his graciously coming to attend my doctoral examination.

There are many people who have played other significant roles along the way. Michael Schiehlen participated in numerous conversations and contributed to refining many of the ideas presented here. I have also had the honor of working closely with Hans Kamp and Gunther Heidemann. Their ideas have undoubtedly influenced those that are contained in this thesis.

I am grateful to my colleagues at Stuttgart University, who will forever remain in my heart and in my thoughts. In particular, I would like to thank: Andre Burkovski, Alexander Fraser, Thomas Müller, Wiltrud Kessler, Lukas Michelbacher, Charles Jochim, Florian Laws, Sabine Dieterle, Andrea Glaser, Patrick Ziering, Arndt Riester, Bernd Bohnet, Hassan Saijad, Nadir Durrani, Alok Kothari, Richard Farkas, Wolfgang Seeker and Sina Zarriß.

Most of all, I would like to express my warmest thanks to my wonderful wife, Shahla, for her unconditional support, patience and invaluable companionship over the years. She deserves more credit than I can put into words. I would also like to thank my dearest children, Datis and Hana, just for being there to give more meaning to my life. Last but not least, I am thankful to my parents, who have always wanted the best for me, and have put my happiness before theirs.

This research is part of SÜKRE project which has been funded by DFG (grant SCHU 2246/4).

Some results of this thesis have been published in the proceedings of international conferences, including among others:

Hamidreza Kobdani and Hinrich Schütze (2010). SUCRE: A Modular System for Coreference Resolution. In Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, Uppsala, Sweden, July 15-16, 2010.

Hamidreza Kobdani and Hinrich Schütze (2011). Supervised Coreference Resolution with SUCRE. In Proceedings of the CoNLL-2011 Shared Task on Modeling Unrestricted Coreference in OntoNotes, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011.

Hamidreza Kobdani, Hinrich Schütze, Andre Burkovski, Wiltrud Kessler and Gunther Heidemann (2010). Relational Feature Engineering of Natural Language Processing. In Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Canada, October 27-29, 2010.

Hamidreza Kobdani, Hinrich Schütze, Michael Schiehlen and Hans Kamp (2011). Bootstrapping Coreference Resolution Using Word Associations. To appear in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, ACL 2011, Portland, Oregon, USA, June 19-24, 2011.

Hamidreza Kobdani, Alexander Fraser and Hinrich Schütze (2009). Word Alignment by Thresholded Two-Dimensional Normalization. In Proceedings of MT Summit XII, Ottawa, Canada, August 26-30, 2009.

Andre Burkovski, Gunther Heidemann, **Hamidreza Kobdani**, and Hinrich Schütze (2010). Methods for coreference visualization and annotation. Technical Report 2010/09, Intelligent Systems Group, University of Stuttgart, 2010.

Contents

1	Introduction	15
1.1	Motivation	15
1.2	Contribution	16
1.3	Framework Overview	17
1.4	Structure	18
2	Literature Review	21
2.1	Introduction	21
2.2	Machine Learning	22
2.3	Statistical Inference	22
2.3.1	Systematic Approach	24
2.3.2	Features and Labels	25
2.3.3	Learner and Model	26
2.4	Supervised Learning	26
2.4.1	Naive Bayes Classifier	28
2.4.2	Maximum Entropy	29
2.4.3	Decision Tree	30
2.5	Unsupervised Learning	32
2.5.1	Partitional Clustering	32
2.5.2	Hierarchical Clustering	33
2.6	Semi-supervised Learning	34
2.6.1	Self-Training	34
2.6.2	Co-Training	35
2.7	Natural Language Processing	36
2.7.1	Tasks	37

CONTENTS

2.8	Conclusion	40
3	Coreference Resolution	41
3.1	Introduction	41
3.2	Problem Formulation	42
3.2.1	Methods	44
3.2.2	Feature Types	45
3.3	Rule-based Models	46
3.3.1	Hobbs	46
3.3.2	CogNIAC	47
3.3.3	Mitkov	49
3.3.4	Multi-Pass-Sieve	51
3.4	Supervised Models	54
3.4.1	Decision Tree	54
3.4.2	Generative Probabilistic Approach	57
3.4.3	Ranking Approach	58
3.5	Unsupervised Models	60
3.5.1	Nonparametric Bayesian Model	60
3.5.2	Markov Logic Model	61
3.6	Semi-supervised Models	62
3.6.1	Self-Training (bootstrapping)	62
3.6.2	Co-Training	64
3.7	Evaluation Metrics	64
3.7.1	MUC	66
3.7.2	B-Cubed	69
3.7.3	CEAF	71
3.7.4	BLANC	73
3.7.5	Evaluation Problem	75
3.8	Conclusion	76
4	Supervised Coreference Resolution	77
4.1	Introduction	77
4.2	System Architecture	78
4.3	Preprocessing	80
4.3.1	Tokenizer	80

CONTENTS

4.3.2	Part of Speech Tagger	81
4.3.3	Number Recognizer	81
4.3.4	Gender Recognizer	81
4.3.5	Semantic Class Recognizer	81
4.3.6	Named Entity Recognizer	82
4.3.7	Parser	82
4.3.8	Markable Detector	84
4.4	Pair Estimation	86
4.4.1	Learning	86
4.4.2	Classification	95
4.5	Chain Estimation	95
4.6	SUCRE vs. Reconcile-2010	97
4.7	SemEval-2010	97
4.7.1	Task	97
4.7.2	System Description	99
4.7.3	Data Sets	100
4.7.4	Feature Sets	101
4.7.5	Results	101
4.7.6	Evaluation Problem	104
4.8	CoNLL-2011	106
4.8.1	Task	106
4.8.2	Data set	107
4.8.3	Scorer Problem	107
4.8.4	Results	109
4.9	Conclusion	110
5	Relational Feature Engineering	112
5.1	Introduction	112
5.2	Related Work	114
5.3	Process Model	114
5.4	Data Model	116
5.5	Relational Feature Engineering	119
5.6	Framework Architecture	121
5.7	Coreference Resolution	122

CONTENTS

5.7.1	System Description	124
5.7.2	Evaluation	133
5.8	CoRe Features in Multiple Languages	133
5.8.1	Feature Engineering	134
5.8.2	Data Sets	134
5.8.3	Results	135
5.9	Conclusion	143
6	Unsupervised Coreference Resolution	145
6.1	Introduction	145
6.2	Feature Spaces	146
6.2.1	Shallow Feature Space	147
6.2.2	Linguistic Feature Space	147
6.3	Case Study: Word Alignment	148
6.3.1	Related Work	150
6.3.2	Associative Information	152
6.3.3	2D-Linking	155
6.3.4	Evaluation	157
6.3.5	Summary	161
6.4	Unsupervised CoRe with A-INF	162
6.4.1	System Architecture	163
6.4.2	Learning	164
6.4.3	Classification	166
6.4.4	Chain Estimation	167
6.4.5	Evaluation	168
6.4.6	Data Sets	168
6.5	Unsupervised CoRe with UNSEL	171
6.5.1	Related Work	172
6.5.2	System Architecture	173
6.5.3	Evaluation	175
6.5.4	Data Sets	175
6.5.5	Results	176
6.6	Conclusion	180

CONTENTS

7	Conclusions and Future Work	182
7.1	Contributions of this Thesis	182
7.2	Future Work	183
7.2.1	Chain-based coreference resolution	183
7.2.2	Self-training for chains	186
7.3	Summary	187
A	Visualization	190
A.1	Introduction	190
A.2	Method	191
A.2.1	Interaction techniques	193
A.2.2	Annotation	194
A.2.3	Feature engineering	195
A.3	Summary	196
B	SemEval-2010 Feature Sets	197
B.1	Catalan	197
B.2	German	200
B.3	English	203
B.4	Spanish	205
B.5	Italian	205
B.6	Dutch	206

List of Tables

4.1	Filter Examples	92
4.2	Compare SUCRE and Reconcile-2010	97
4.3	Results of SUCRE	102
4.4	Results of best systems without SUCRE	103
4.5	Scores of the baselines	105
4.6	SUCRE official results at CoNLL-2011 (bugged scorer)	108
4.7	SUCRE official results at CoNLL-2011 (fixed scorer)	108
4.8	SUCRE results at CoNLL-2011 (system markables)	109
4.9	SUCRE results at CoNLL-2011 (true markables)	110
5.1	Relational Data Model: Part 1	127
5.2	Relational Data Model: Part 2	128
5.3	Variable set V	129
5.4	Terminal set T	129
5.5	Grammar	130
5.6	Results for the identical features	137
5.7	Scores of the baselines	137
5.8	Relative scores of the noun type features	138
5.9	Relative scores of the pronoun type features	139
5.10	Relative scores of the grammatical features	141
5.11	Relative scores of the agreement features	142
6.1	Association values of the English word this	154
6.2	Example for indirect associations	157
6.3	The association-score matrix M for the translation example	157
6.4	The decision matrix D for the translation example	158

LIST OF TABLES

6.5	Alignment computed by competitive linking	158
6.6	Results of heuristic methods and IBM Model 1	160
6.7	Results of the different combinations of heuristic methods	161
6.8	Results of A-INF and comparable coreference systems	170
6.9	Scores of MSCORE and comparable coreference systems	177
6.10	Compare SUCRE and UNSEL on OntoNotes	179

List of Figures

2.1	Information Processing System	24
2.2	Learning System	24
2.3	Basic form of Self-Training	35
2.4	Basic form of Co-Training	36
3.1	Bootstrapping Coreference Classifiers	63
3.2	Feature selection for Co-Training	65
3.3	Example: key partition	68
3.4	Example: response partition 1	68
3.5	Example: response partition 2	68
3.6	Example: response partition 3	68
4.1	Top Level Architecture of SUCRE	78
4.2	Preprocessing Pipeline	80
4.3	Most frequent parse tree paths in OntoNotes	83
4.4	Noun Phrases in Phrase Structures	85
4.5	Nested Noun Phrases in Phrase Structures	85
4.6	Markable Detection from Phrase Structures	87
4.7	Noun Phrases in Dependency Structures	88
4.8	Detected Markables	88
4.9	Markable Detection from Dependency Parse Information	89
4.10	Pair Estimation in SUCRE: Learning	90
4.11	Backward and Forward Pair Generation Algorithms	91
4.12	Decision Tree	93
4.13	Pair Estimation in SUCRE: Classification	94
4.14	Chain Estimation Algorithm	96

LIST OF FIGURES

6.1	Preprocessing	148
6.2	Shallow vs. linguistic (Rich) features	149
6.3	Pair Estimation in A-INF: Learning	164
6.4	Pair Estimation in A-INF: Classification	166
6.5	Chain Estimation	168
6.6	MUC learning curve for A-INF	171
6.7	System Architecture of UNSEL	173
6.8	Self-Training in UNSEL	174
7.1	Sampling algorithm	185
A.1	A subset of the visualizations	192

Chapter 1

Introduction

1.1 Motivation

Coreference is one of the central topics in the study of natural language. It has been researched in a wide range of disciplines, among them theoretical, corpus and computational linguistics. Coreference information is also beneficial in a number of natural language processing applications, including machine translation, automatic abstracting, information extraction and question answering. As a result, coreference resolution is playing an increasingly important role in many research fields, and has enjoyed increased interest in recent years.

Considerable engineering effort is needed to create a coreference resolution system, and a modular approach to the problem can reduce the time and effort needed to create a system. But a considerable part of the effort needed concerns feature engineering, and a systematic approach to the feature engineering not only increases the modularity but can also help in a number of ways: (i) to identify, appraise, select and synthesize all high quality features relevant to a specific NLP task; (ii) to increase portability of features between systems and tasks by reusing the existing features instead of creating new features when moving system from a domain, language or dataset to another domain, language or dataset; (iii) to have a uniform approach to the features in different modes of machine learning, i.e. supervised, unsupervised and semisupervised; (iv) to have a uniform definition of hard constraint (filters) and soft constraint (features); and (v) to make it possible to work in shallow and rich feature spaces.

The lack of sufficient labeled data makes our system much more attractive in that it is able to learn in different modes of machine learning (i.e. supervised, unsupervised and semisupervised).

1.2 Contribution

The major contributions of this thesis can be summarized as follows:

- a modular supervised system for coreference resolution
 - increases the extent to which a system is composed of separate, interchangeable components by breaking down system into modules.
 - has the advantage that each module accomplishes one function and contains everything necessary to accomplish this.
 - improves maintainability by enforcing logical boundaries between components.
 - reduces the time and effort needed to create a coreference resolution system.
- a novel approach for feature engineering of natural language processing that is based on representing unstructured text in a relational data model. This leads to:
 - More coherency of the methods and algorithms that are needed in different stages by presenting a common structure of the text corpus based on the relations between the textual entities.
 - A natural formalism for supporting the definition and extraction of features.
 - It can be employed to reduce the size of the feature vector (in the feature space of an NLP task).
 - Its modular approach makes it possible to use it with other machine learning tools.
 - It is possible to apply many existing methods of knowledge discovery in database to the text corpus with less design and implementation effort.

- a modular framework supports three major modes of machine learning, namely, (i) Supervised; (ii) Unsupervised; and (iii) Self-Trained.
 - I will demonstrate that lexical information can be used to develop an unsupervised model for shallow coreference resolution.
 - I will also introduce an unsupervised self-trained method that takes a two-learner two-feature-space approach, in which the two learners are supervised and unsupervised. The feature spaces are the shallow and rich feature spaces.
 - I will also be showing that the performance of an unsupervised self-trained system is better than the performance of other unsupervised systems when it is self-trained on the automatically labeled corpus and uses the leveraging effect of rich linguistic features.
 - A flexible and modular framework is able to learn from data with different quality and domain. Not only is it able to deal with shallow information spaces, but it can also deliver competitive results for rich feature spaces.

1.3 Framework Overview

My experiments were conducted using the M`CORE` system (“Modular C`O`reference R`E`solution”). M`CORE` can operate in three different settings: unsupervised, supervised, and self-trained. The supervised subsystem utilized was *SUCRE* (“S`U`pervised C`O`reference R`E`solution”). It is trained on a labeled corpus (manually or automatically labeled), similar to standard CoRe systems. The unsupervised subsystem used was *A-INF* (“A`SS`ociation I`N`formation”). It uses the association scores between heads as the distance measure when clustering markables. Finally, the unsupervised self-trained subsystem UNSEL (“UN`S`upervised S`E`LF-trained”) uses the unsupervised subsystem A-INF to automatically label an unlabeled corpus that is then used as a training set for the supervised subsystem *SUCRE* (“S`U`pervised C`O`reference R`E`solution”).

In summary, I have developed **M`CORE`**, a framework for coreference resolution. It contains the following subsystems:

1. **SUCRE**: a pair-based supervised system which is trained on a labeled corpus (manually or automatically labeled), similar to standard coreference resolution systems.
2. **A-INF**: an unsupervised system that uses the association scores between heads as the distance measure when clustering markables.
3. **UNSEL**: an unsupervised self-trained system. It uses the unsupervised subsystem A-INF to automatically label an unlabeled corpus that is then used as a training set for SUCRE.

1.4 Structure

This thesis is structured into 7 chapters.

Chapter 2 presents an overview of the theories and the methods of Machine Learning and Natural Language Processing. The foundational background will be presented and the basics of different approaches to machine learning reviewed. The approaches which will be discussed include (i) supervised; (ii) unsupervised; and (iii) semi-supervised. Concerning supervised approaches, three well-known methods will be reviewed : naive Bayes, maximum entropy and decision tree. For unsupervised learning, two types of clustering will be considered: partitional and hierarchical. Two commonly used semi-supervised methods are Self-Training and Co-Training, which will be presented from a semi-supervised approach. Finally a very short introduction to the field of natural language processing and some related tasks will be presented. In its entirety this Chapter forms the necessary foundations related to the proposed framework.

Chapter 3 reviews the literature and related work concerning coreference resolution and presents several techniques that are used for coreference resolution. First, the problem of coreference resolution will be formulated. Afterward the review of the literature continues in four directions: (i) Rule-based approaches, where Hobbs [Hobbs, 1986], CogNIAC [Baldwin, 1996], Mitkov [Mitkov, 1998] and Multi-Pass-Sieve [Raghunathan et al., 2010] methods will be presented; (ii) Supervised approaches, where decision tree [Soon et al., 2001], generative probabilistic [Ge et al., 1998] and ranking approaches [Denis and Baldridge, 2007a] are discussed; (iii) Unsupervised approaches, where a nonparametric Bayesian model

[Haghighi and Klein, 2007] and a Markov logic model [Poon and Domingos, 2008] will be presented; and finally (iv) Semi-supervised, where two methods of Self-Training [Ng and Cardie, 2003] and Co-Training [Müller et al., 2002] for coreference resolution will be presented.

This chapter also presents the most commonly used evaluation metrics for coreference resolution. This part is accompanied by a running example and at the end will discuss the problems concerning these metrics.

In **Chapter 4**, a supervised system (SUCRE) for coreference resolution will be proposed. SUCRE is multilingual and has a modular architecture. The modular architecture of SUCRE contains three main modules: preprocessing, pair estimation and chain estimation. The details of these modules will be presented by breaking them down into separate interchangeable modules in order to show how much strongly modular this approach is.

In this chapter the successful participation of SUCRE in two international shared tasks on coreference resolution will also be reported on, namely, (i) SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010]; and (ii) CoNLL-2011 Shared Task on Modeling Unrestricted Coreference in OntoNote [Pradhan et al., 2011b]. I see the good performance of SUCRE in the shared tasks as a demonstration of the strength of the system I propose and general validity of my design. The results of SUCRE show the promise of my framework.

Chapter 5 presents a new framework for feature engineering of natural language processing (NLP) that is based on representing unstructured text in a relational data model. It includes powerful and flexible methods for implementing and extracting new features, thereby reducing the time and effort needed to create a natural language processing system.

To demonstrate the utility of my framework in NLP feature engineering, an instantiation and evaluation of the framework for the problem of coreference resolution in multiple languages will be presented. This instantiation represents the data model of SUCRE and the details of its relational feature engineering approaches will be discussed that were not presented in the previous chapter.

This chapter also presents a relational feature definition language, which can be used instead of SQL for relational feature definition. Roughly speaking, it embeds all the needed inner joins in SQL to provide a fluent access to the attributes

of different relations in the data model.

In the last part of this chapter, the relative contributions of the various feature types for coreference resolution of Dutch, German, Italian and Spanish will be investigated and reported. The aim is to define a common feature set that can be used for these different languages.

Chapter 6 starts by presenting a feature space model of natural language processing to provide a better understanding of the possible features. This feature space model contains two feature spaces: shallow space, with only one term-frequency-based feature; and rich space, which is n-dimensional. Using the shallow feature space, a case study consisting of a novel method for word alignment will be presented. This shallow feature space also has been used for a new unsupervised approach to coreference resolution (A-INF) that uses lexical association scores between heads as the distance measure when clustering markables.

In the last part of this chapter, a novel unsupervised self-training method will be presented that first labels a corpus using the unsupervised model A-INF and then trains the supervised model SUCRE on this automatically labeled training corpus. Even though I train on a labeled corpus, the labeling of the corpus is produced in a completely automatic fashion, without recourse to human labeling. Thus, it is an unsupervised approach.

In **Chapter 7**, this work will be concluded with a review of the contributions of this proposed framework along with considerations for future work.

Appendix A presents a brief introduction to a visualization method of coreference data based on Self Organizing Maps (SOMs). This method has been designed and implemented as part of the main project SÜKRE. M-CORE framework is a part of SÜKRE.

Appendix B contains the feature sets of six languages that were used in SUCRE for SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010]

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the literature concerning Machine Learning and Natural Language Processing. I present the foundational background and review the basics of different approaches to machine learning. The approaches which will be discussed include (i) supervised; (ii) unsupervised; and (iii) semi-supervised. Concerning supervised approaches, I concisely review three well-known methods: naive Bayes, maximum entropy and decision tree. For unsupervised learning, I consider two types of clustering: partitional and hierarchical. Two commonly used semi-supervised methods are Self-Training and Co-Training which are presented concerning semi-supervised approaches. Finally I will present a very short introduction to the field of natural language processing and some related tasks. In its entirety this Chapter forms the necessary foundations related to the framework and methods proposed in this thesis.

This chapter is organized as follows. Section 2.2 presents a short introduction to machine learning. Statistical inference is described in Section 2.3. Sections 2.4, 2.5 and 2.6 introduce supervised, unsupervised and semi-supervised approaches. Section 2.7 presents a short description of natural language processing. And finally, Section 2.8 concludes this chapter.

2.2 Machine Learning

The term “learning” has come to be used to refer to the process of acquiring new or modifying existing knowledge, behaviors, skills, values, or preferences. The ability to learn is possessed by humans, animals and some machines.

Machine Learning is a generic term for the artificial creation of knowledge from experience: An artificial system learns from examples to recognize complex patterns and make intelligent decisions. That is, it learns not just the examples, but it also recognizes laws in the examples (training data). Machine Learning is a central topic in the field of Artificial Intelligence. Nilsson [1996] has defined it as: “A machine learns whatever changes its structure, program or data (based on its inputs or in response to external information) in such manner that its expected future performance improves.”

The machine-learning algorithms are often, although not always, grounded in statistical inference to automatically learn the rules through the analysis of input instances. In the following we will take a look at statistical inference.

2.3 Statistical Inference

Statistical inference concerns inferring conclusions based on data. In other words it is about approximating the unknown distribution by looking at the data generated by that distribution. There are several different approaches to computational learning theory.

Two main reasons for these differences are (i) the different definitions of probability; and (ii) the different assumptions on the generation of samples.

The different definitions of probability that lead to two different inference approaches include:

Frequentist inference: Frequentist inference uses the frequency interpretation of probability (i.e. the sampling distribution of a statistic), where any given experiment can be considered as one of an infinite sequence of possible repetitions of the same experiment [Everitt, 2002]. In other words, by looking at the repeated sample, the frequentist properties of any statistical inference procedure can be described.

Bayesian inference: Bayesian inference is based on the probability that a particular hypothesis is true given some observed evidence. The key idea is that the probability of an event A given an event B depends not only on the relationship between events A and B but also on the marginal probability of occurrence of each event:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \quad (2.1)$$

Statistical inference of a hypothesis most often includes the following components:

Statistical Model: it models the random process that is supposed to generate the data, and describes how one or more random variables are related to one or more random variables.

Data Set: it is a particular realization of the random process. It comes from actual observations that are performed by sampling a statistical population. Each observation measures one or more attributes (i.e. features) of independent objects or individuals.

A statistical model includes a set of assumptions that describes the data generation from the observed data. There are three levels of modeling assumptions, as follows [Cox, 2006, van der Vaart, 1998]:

Fully-parametric: the data-generation process is described by a probability distribution involving only a finite number of unknown parameters. For example, a distribution of population values is described as truly Normal, with unknown mean and variance, and the data is generated by *simple* random sampling.

Non-parametric: there are fewer data-generation process assumptions (i.e. they are minimal) in comparison with the fully parametric models. For example, a continuous probability distribution can be estimated using the sample median when the data is generated by *simple* random sampling.

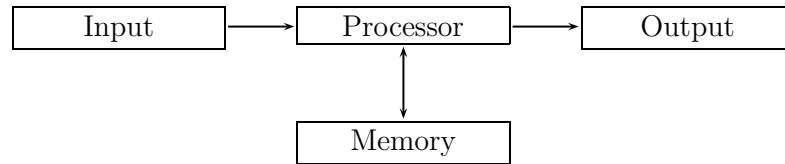


Figure 2.1: Information Processing System

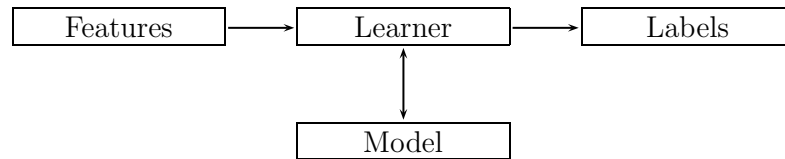


Figure 2.2: Learning System

Semi-parametric: the data-generation process assumptions are between fully-parametric and non-parametric approaches. For example, population distribution can be estimated using a finite mean and the parametric assumption that the mean response level in the population is a linear function of some covariate.

2.3.1 Systematic Approach

From a systematic point of view, an artificially intelligent machine (i.e. it is able to learn) includes an information processing system.

Figure 2.1 presents a general model of an information processing system that is made up of four basic parts: **(i) Input:** the data received by the system; **(ii) Processor:** processing of the data; **(iii) Memory:** retention of the data; and finally **(iv) Output:** the data sent from the system.

Analogue to the general model presented in Figure 2.1, a general model of a learning system is presented in Figure 2.2, where the input data are the features and the output are the labels. A learner is a processor (i.e. procedure) that keeps what it learns in the memory as a model.

2.3.2 Features and Labels

Features are the measurable properties of data objects. They are produced in the preprocessing step. A data object can be a record, a point, an event, or a textual entity [Tan et al., 2005]. A set of data objects is called a “data set”.

An attribute is a property or characteristic of a data object that generally falls into one of two types:

Qualitative (categorical): where an attribute is described in terms of some quality or categorization. Qualitative data are often referred to as categorical data (i.e. they can be placed into distinct categories). No arithmetic operations can be applied to them. They sometimes can be arrayed in a meaningful order. For example, assignment evaluation (i.e. fail, pass, good, excellent) is qualitative.

Quantitative (numeric): where an attribute is described with a numerical scale. Quantitative data have a numeric nature. Arithmetic operations can be applied to them, and they can be arrayed in a meaningful order. They are classified into two groups: (i) discrete, those that can be counted; and (ii) continuous, those including any possible value within their value range.

A measurement scale is a function that associates a symbolic or numerical value with an attribute of a data object. The output of a measurement scale is a feature value. In other words, a measurement scale is the way we define a feature. Different types of measurement scales are:

Nominal: applicable on qualitative attributes. It implies distinctness. The possible operations are $=$ and \neq .

Ordinal: applicable on qualitative attributes. It implies order. The possible operations are $<$, \leq , $>$ and \geq .

Interval: applicable on quantitative attributes. It implies addition. The possible operations are $+$, $-$.

Ratio: applicable on quantitative attributes. It implies multiplication. The possible operations are \times and \div .

2.3.3 Learner and Model

Different algorithms of statistical machine learning can be categorized into these main types:

Supervised Learning: The algorithm learns a function from given pairs of input and output (labeled data). In other words, it generates a function that maps inputs to desired outputs.

Unsupervised Learning: The algorithm generates a model for a set of inputs (unlabeled data). The main problem of this kind of learning is partitioning the input set into subsets in such way that each subset can be handled with an appropriate function. One example of using unsupervised learning is data compression, where the probability distribution of the input set plays an important role.

Semi-supervised Learning: The algorithm combines both labeled and unlabeled data to generate an appropriate function or a model.

Reinforcement Learning: In Reinforcement Learning the aim is the maximization of rewards given as feedback on the actions which are performed in an environment by the learning agent. In other words [Sutton and Barto, 1998]: *“Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal.”*

This chapter will review the Supervised, Unsupervised and Semi-supervised methods in more detail.

2.4 Supervised Learning

The problem of supervised learning can be formulated as follows [Duda et al., 2001]: Given an unknown function $g : \mathcal{X} \rightarrow \mathcal{Y}$, which is known as the ground truth and maps input instances $\mathbf{x} \in \mathcal{X}$ to output labels $y \in \mathcal{Y}$, along with training data $\mathbf{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, the goal is to find a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that approximates as closely as possible the correct mapping g . In decision theory, the condition “approximates as closely as possible” is defined by specifying a

loss function that assigns a specific value to “loss” resulting from producing an incorrect label. The goal then is the expected loss minimization. The distribution of \mathcal{X} and the ground truth function $g : \mathcal{X} \rightarrow \mathcal{Y}$ can only be computed empirically by providing a large number of samples of \mathcal{X} and their corresponding labels \mathcal{Y} . The type of label being predicted determines the particular loss function needed. For example, in the case of binary classification, a simple zero-one loss function would be sufficient. The zero-one loss function corresponds to assigning a loss of 1 to any incorrect labeling. In other words it is equivalent to computing the accuracy of the classification procedure over a set of test data where the goal is to maximize this accuracy.

From a probabilistic point of view, the supervised learning problem is to estimate a function f which is the probability of each possible output label y given a particular input instance x :

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = f(\mathbf{x}; \boldsymbol{\theta}) \quad (2.2)$$

where the feature vector input is x , and the function f is typically parametrized by some parameters θ . There are two approaches to the problem:

1. Discriminative approach, where the function f is estimated directly;
2. Generative approach, where the inverse probability $p(x|y)$ is estimated and combined with the prior probability $p(y|\boldsymbol{\theta})$ using Bayes’ rule, as follows:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}|y)p(y|\boldsymbol{\theta})}{\sum_{y' \in \mathcal{Y}} p(\mathbf{x}|y')p(y'|\boldsymbol{\theta})} \quad (2.3)$$

In the case that the labels are continuously distributed¹, the denominator will be an integration:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}|y)p(y|\boldsymbol{\theta})}{\int_{y' \in \mathcal{Y}} p(\mathbf{x}|y')p(y'|\boldsymbol{\theta}) \, dy'} \quad (2.4)$$

The following supervised learning methods belong to the set of most commonly applied machine learning methods.

¹If the output of a supervised learning algorithm is discrete, it is called a classifier; if continuous, it is called a regressor.

2.4.1 Naive Bayes Classifier

A Naive Bayes classifier is based on Bayes' theorem (see Equation 2.1), with the naive assumption that all features are independent. For a more descriptive term of the underlying probability model, we can call it the “independent feature model”. A naive Bayes classifier assumes that a particular feature has no relation with any other feature.

Given a set of features $\{F_1 \dots F_n\}$, the probability model for a classifier is a conditional model of the form

$$p(C|F_1, \dots, F_n) \tag{2.5}$$

where the class variable C , depends on features $\{F_1 \dots F_n\}$. Using Bayes' theorem, we write

$$p(C|F_1, \dots, F_n) = \frac{p(C) p(F_1, \dots, F_n|C)}{p(F_1, \dots, F_n)} \tag{2.6}$$

As the denominator does not depend on C and the features values are given, the denominator is constant and can be deleted from the model. We are then only interested in the numerator, which can be written as follows:

$$p(C) p(F_1|C) p(F_2|C, F_1) p(F_3|C, F_1, F_2) \dots p(F_n|C, F_1, F_2, F_3, \dots, F_{n-1}) \tag{2.7}$$

By taking the *naive* conditional independence assumptions into account (i.e. assuming that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$), it is equal to:

$$p(C) p(F_1|C) p(F_2|C) p(F_3|C) \dots p(F_n|C) \tag{2.8}$$

This means that the desired conditional distribution can be expressed as:

$$p(C|F_1, \dots, F_n) \simeq p(C) \prod_{i=1}^n p(F_i|C) \tag{2.9}$$

This is a naive Bayes model. Models of this kind are much more manageable, because they factor into independent probability distributions $p(F_i|C)$.

2.4.2 Maximum Entropy

Maximum Entropy modeling has been successfully applied to many fields such as Computer Vision, Spatial Physics and Natural Language Processing. Maximum Entropy models can be used for learning from many heterogeneous information sources [Manning and Schütze, 1999]. Maximum entropy classifiers are considered alternatives to Naive Bayes classifiers, because they do not require the naive assumption of the independent features. However, they are significantly slower than Naive Bayes classifiers, and thus may not be appropriate when there is a very large number of classes.

A typical binary maximum entropy classifier can be implemented using logistic regression. Logistic regression can be explained with the logistic function as follows:

$$L_f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}} \quad (2.10)$$

where the variable z is defined as:

$$z = \beta_0 + \beta_1 y_1 + \beta_2 y_2 + \beta_3 y_3 + \cdots + \beta_k y_k \quad (2.11)$$

where β_0 is called the “intercept”, and $\beta_1, \beta_2, \dots, \beta_k$ are called the “regression coefficients” of features y_1, y_2, \dots, y_k respectively. Logistic regression describes the relationship between one or more independent features and a binary label, expressed as a probability.

Generally, maximum entropy classifiers are implemented using the principle of maximum entropy [Cover and Thomas, 1991]. Suppose we have some testable label L about a quantity x taking values in x_1, x_2, \dots, x_n where x_i is a feature vector of size m . This label can be expressed by m constraints on the expectations of the feature values; that is, the following probability distribution must be satisfied, where $f_k(x_i)$ returns the k th element of the feature vector x_i :

$$\sum_{i=1}^n p(x_i|L) f_k(x_i) = C_k \quad k = 1, \dots, m. \quad (2.12)$$

In addition, the probabilities must sum to one.

$$\sum_{i=1}^n p(x_i|L) = 1. \quad (2.13)$$

The probability distribution with maximum information entropy subject to above constraints is:

$$p(x_i|L) = \frac{e^{\lambda_1 f_1(x_i) + \dots + \lambda_m f_m(x_i)}}{\sum_{i=1}^n e^{\lambda_1 f_1(x_i) + \dots + \lambda_m f_m(x_i)}} \quad (2.14)$$

The λ_k parameters are Lagrange multipliers whose particular values are determined by the constraints according to

$$C_k = \frac{\partial}{\partial \lambda_k} \log \sum_{i=1}^n e^{\lambda_1 f_1(x_i) + \dots + \lambda_m f_m(x_i)} \quad k = 1, \dots, m. \quad (2.15)$$

The above equation presents m simultaneous equations, which are usually solved by numerical methods.

2.4.3 Decision Tree

A decision tree is an efficient non-parametric method for supervised learning. It includes a hierarchical data structure with a divide-and-conquer algorithm, where a problem is recursively broken down into two or more sub-problems of the related type, until all sub-problems are simple enough to be solved directly.

Generally, decision trees have the following advantages over many other methods of machine learning [Rokach and Maimon, 2008]:

Versatility: they can be used for a wide range of machine learning tasks, such as classification, regression, clustering and feature selection.

Self-explanatory: they are easy to follow and understand.

Flexibility: they are able to handle different types of input data (e.g. nominal, numeric and textual).

Adaptability: they can process datasets containing errors or missing values.

High predictive performance: with a relatively small computational effort they usually perform very well.

The decision tree is a directed tree with a node called a root (i.e. it has no incoming edges) [Rokach and Maimon, 2008]. All other nodes have exactly one incoming edge and zero or more outgoing edges. A node with outgoing edge is called “internal” or test node. A node without outgoing edge is called “leaf” (also known as terminal or decision node).

The decision tree partitions the instance space according to the features value. For this purpose it chooses one feature of the data at each node that most effectively splits the corresponding instance space. There are different measures that can be used for splitting the instance space. Two frequently used measures are Gini index and information gain, which are explained as follows:

Gini index: Gini index for a given feature f is:

$$GINI(f) = 1 - \sum_C (p(C|f))^2 \quad (2.16)$$

where $p(C|f)$ is the relative frequency of class C and feature f . The Gini index is minimum (i.e. 0.0) when the feature corresponds to only one class, implying the most interesting information. The Gini index reaches its maximum (i.e. $1 - \frac{1}{|C|}$) when the feature is equally distributed among all classes, implying the least interesting information.

The Gini index cannot measure the quality of split of each feature (i.e. at a node of tree). A metric that measures this quality is $GINI_{split}$, which is defined as:

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n_j} GINI(i) \quad (2.17)$$

where n_i is the number of instances at child node i , and n_j is the number of instances at the parent node.

Information gain: The change in entropy from a prior state to a state that takes some information is the expected information gain [Mitchell, 1997]:

$$IG(f) = H(C) - H_f(C) \quad (2.18)$$

where f is the feature value, C its corresponding class, and entropy is defined as follows:

$$H(C) = - \sum_i P(C_i) \log_2 P(C_i) \quad (2.19)$$

$$H_f(C) = \sum_f \frac{|C_f|}{|C|} H(C_f) \quad (2.20)$$

As for the Gini index, if a feature takes a large number of distinct values, the information gain would not be a good measure of the quality of split. In such cases the information gain ratio is used instead. The information gain ratio for a feature is calculated as follows:

$$IGR(f) = \frac{IG(f)}{SInf(C)} \quad (2.21)$$

$$SInf(C) = - \sum_i \frac{|C_i|}{|C|} \log_2 \frac{|C_i|}{|C|} \quad (2.22)$$

2.5 Unsupervised Learning

The problem of unsupervised learning can be formulated as follows: Given a set of observed data, the goal is to estimate an unobservable underlying probability density function. This definition is reminiscent of density estimation in statistics. However, unsupervised learning consists of many other methods that try to summarize and explain key features of the data.

Clustering is one of the key ideas in unsupervised learning. It is the assignment of a set of data (observations) into subsets (clusters), so that data in the same cluster are similar in some sense. For example, one can find that most objects in the natural world fall into one of two categories: things that are brown and run away, and things that are green and don't run away. The first group is called animals, and the second, plants. Clustering is the operation of grouping things together [MacKay, 2003].

There are several different types of clustering that generally fall into the two following types.

2.5.1 Partitional Clustering

Partitional clustering aims to determine all clusters at once. It is simply a division of the set of data objects into non-overlapping subsets [Tan et al., 2005]. An example of partitional clustering is K-means Clustering, which is presented below.

K-means Clustering is a parametric approach that aims to partition n data objects into k clusters where each data object is assigned to the cluster with the nearest mean. In other words, it tries to find the centers of natural clusters of the data objects.

To provide a formal description, consider a set of data objects (x_1, x_2, \dots, x_n) , where each data object is a real vector. K-means clustering aims to partition the n data objects into k clusters ($k \leq n$) $C = C_1, C_2, \dots, C_k$, so that the within-cluster sum of squares is minimum:

$$\arg \min_{\mathbf{C}} \sum_{i=1}^k \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 \quad (2.23)$$

where μ_i is the mean of data objects in C_i . An algorithm that is often used for k-means clustering has the following steps, which are iterated (counted by t) until a convergence is reached (i.e. the assignments do not longer change) [MacKay, 2003]:

Assignment: assign each data object to the cluster with the closest mean:

$$C_i^{(t)} = \left\{ \mathbf{x}_j : \|\mathbf{x}_j - \mu_i^{(t)}\| \leq \|\mathbf{x}_j - \mu_{i^*}^{(t)}\| \text{ for all } i^* = 1, \dots, k \right\}$$

Update: calculate the new means to be the centroid of the data objects in the cluster:

$$\mu_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j$$

The initialization can be done, for example, by randomly choosing k data objects from the data set as the initial means.

2.5.2 Hierarchical Clustering

Hierarchical clustering is a non-parametric approach that aims to build a hierarchy of clusters. Approaches to hierarchical clustering generally fall into two types:

Agglomerative: a “bottom up” approach, in which one starts by inserting each data object into its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

Divisive: a “top down” approach, in which one starts by inserting all data objects in one cluster, and splits are performed recursively as one moves down the hierarchy.

In order to determine which clusters should be merged, or where a cluster should be split, a measure of similarity/dissimilarity (e.g. a measure of distance) is needed between sets of data objects. Some commonly used measures of distance for hierarchical clustering are (where a and b are two clusters; and a_i and b_i are the data objects of the corresponding clusters):

- Euclidean distance: $\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
- Squared Euclidean distance: $\|a - b\|_2^2 = \sum_i (a_i - b_i)^2$
- Manhattan distance: $\|a - b\|_1 = \sum_i |a_i - b_i|$
- Maximum distance: $\|a - b\|_\infty = \max_i |a_i - b_i|$
- Mahalanobis distance: $\sqrt{(a - b)^\top S^{-1} (a - b)}$ where S is the covariance matrix.

2.6 Semi-supervised Learning

Semi-supervised learning falls between supervised learning and unsupervised learning. It has been shown that unlabeled data, when used in conjunction with a small amount of labeled data, can considerably improve learning accuracy [Abney, 2008]. Labeled data is often generated by a skilled human agent who manually classifies training examples. The cost of labeling process is sometimes unaffordable, whereas generation of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be a better approach. Two commonly used semi-supervised methods are Self-Training and Co-Training. Both are presented below.

2.6.1 Self-Training

Self-Training (also known as Bootstrapping) is probably the oldest semi-supervised learning method. The idea behind Self-Training is simple: use a seed set of labeled data to label a set of unlabeled data, which can be partly added in turn

procedure *SelfTrain*(L_0, U)

1. L_0 is seed labeled data, L is labeled data
2. U is unlabeled data
3. *classifier* \leftarrow *train*(L_0)
4. $L \leftarrow L_0 + \textit{select}(\textit{label}(U, \textit{classifier}))$
5. *classifier* \leftarrow *train*(L)
6. if no stopping criterion is met, go to step 4
7. return *classifier*

Figure 2.3: Basic form of Self-Training [Abney, 2008].

to the set of labeled data for retraining. The process may be continued until a stopping condition is reached.

Figure 2.3 shows a basic form of Self-Training [Abney, 2008]. The *train* function is a supervised classifier that is called *base learner*. It is assumed that the classifier (base learner) makes confident weighted predictions. The *select* function selects those instances where it is most confident.

One simple method that can be used as a stopping criterion is to repeat the steps 4-6 for a fixed, arbitrary number of rounds. Another simple method is to keep it running until convergence, i.e. until the labeled data and classifier stop changing.

2.6.2 Co-Training

Co-training was introduced by Blum and Mitchell [1998]. It is a semi-supervised method that requires two views of the data. In other words, it assumes that each instance is described using two different feature sets. The two feature sets are to provide different, but complementary information about the instances.

Figure 2.4 shows a basic form of Co-Training [Abney, 2008]. As in Self-Training, the *train* function is a supervised classifier with the assumption that

```
procedure CoTrain( $L, U$ )  
  1.  $L$  is labeled data  
  2.  $U$  is unlabeled data  
  3.  $P \leftarrow$  random selection from  $U$   
  4.  $f_1 \leftarrow$  train(view1( $L$ ))  
  5.  $f_2 \leftarrow$  train(view2( $L$ ))  
  6.  $L \leftarrow L +$  select(label( $P, f_1$ )) + select(label( $P, f_2$ ))  
  7. Remove the labeled instances from  $P$   
  8.  $P \leftarrow P +$  random selection from  $U$   
  9. if no stopping criterion is met, go to step 4
```

Figure 2.4: Basic form of Co-Training [Abney, 2008].

it makes confident weighted predictions. Also the *select* function selects those instances where it is most confident. The stopping criteria that were discussed for Self-Training are applicable for Co-training without modification.

The above mentioned SelfTrain and CoTrain procedures suggest strong similarities between co-training and self-training. If the classifier pair (f_1, f_2) is considered a classifier with an internal structure, then Co-training can be seen as a special case of Self-Training [Abney, 2008].

2.7 Natural Language Processing

Natural language processing (NLP) is a field of computer science that focuses on human (natural) languages. NLP has significant overlap with the field of computational linguistics, and is often considered a sub-field of artificial intelligence.

NLP can deal with many tasks. For example, natural language understanding (an NLP task) can be considered as an AI-complete problem that its difficulty is equivalent to solving the central artificial intelligence problem (i.e.

making computers as intelligent as people), because in order to solve it we require extensive knowledge of the outside world and the ability to manipulate it [Charniak and McDermott, 1985].

Most of the recent NLP algorithms are based on machine learning methods and usually use statistical inference. The approaches currently taken in NLP require an understanding of a number of disparate fields, including linguistics, computer science, and statistics.

As described above, machine learning is the basis of the modern approaches to natural language processing. There are many systems that are based on large sets of hand-produced rules. In the machine learning approach, a corpus that typically consists of a set of documents is used to train a system. The most important advantages of systems that take machine-learning approaches over systems that use hand-produced rules are:

- The learning procedure can automatically focus on the most common cases. This greatly reduces the effort needed to find such common cases. It is not an easy task for humans to manually process a very large amount of data to extract the rules.
- Using statistical inference, it is possible to develop algorithms that are robust to unseen and erroneous input. Generally, systems with hand-written rules have very poor performance in such cases.
- More input data to machine learning systems can make them more accurate. However, in order to achieve more accurate hand-written based systems we have to increase the complexity of the rules, which is a much more difficult task.

2.7.1 Tasks

The following list contains some of the NLP tasks that, to a greater or lesser extent, are related to the research context of this dissertation. Some of these tasks have real-world applications, and others are used as prerequisites of larger ones.

Part-of-speech (POS) tagging

This is the task of determining the part of speech for each word. It is not a trivial task (i.e. it is harder than just having a list of words and their parts of speech) because there are many words, especially common ones, that have multiple parts of speech. For example, the word "book" can be a noun (e.g. in "the book on the table") or can be a verb (e.g. in "to book a flight"). The amount of such ambiguity differs from language to language. The less inflectional morphology a language has, the more such ambiguity arises (e.g. English is particularly prone to such ambiguity) [Charniak, 1997b].

Named entity recognition (NER)

Given a stream of text, this is the task of determining which items in the text map to predefined categories such as the names of persons, organizations, locations, quantities, etc. For NER both linguistic grammar-based techniques as well as statistical models are used.

Text Chunking

This is the task of dividing a text into syntactically correlated parts of words.

For example, the sentence "I heard a new president will be elected in October" can be chunked as: [NP I] [VP heard] [NP a new president] [VP will be elected] [PP in] [NP October] . Text chunking can be considered as an intermediate step towards full parsing.

Parsing

This is the process of analyzing a text to determine its grammatical structure with respect to a given grammar. Since the grammars of natural languages are ambiguous, typical sentences have multiple possible analyse, i.e. a typical sentence can have thousands of potential parses.

Machine translation (MT)

This is the task of translating text or speech from one natural language to another. At its easiest level, it performs simple word-to-word translation, but that alone

usually produces a very poor translation of a text. In order to produce good translations, different types of knowledge that humans possess (e.g. grammar and semantics) are needed.

Coreference resolution (CoRe)

This is the process of finding noun phrases (markables) referring to the same real world entity or concept. Anaphora resolution is a specific example of this type of task. Specifically, CoRe is the process of finding noun phrases (markables) to which pronouns refer.

Automatic Summarization

This concerns the production of a readable summary (a shortened version) of a text. There are different types of summaries, including generic summaries and query-relevant summaries.

Question answering (QA)

Given a human-language question, this is the task of determining its answer. There is a wide range of question types, including, for example, the questions with a specific right answer such as “Where is the capital of Germany?”, or open-ended questions such as “What is the meaning of beauty?”. The typical approaches to this task use either a pre-structured database or a collection of natural language documents (a text corpus).

Relationship extraction

This is the task of detecting semantic relationships among a set of textual entities (e.g. who is the father of whom).

Sentiment analysis

Given a set of documents, this is the task of extracting subjective information (i.e. determining “polarity” about specific objects). The documents are usually online reviews. Sentiment analysis is used especially for identifying public opinion trends in social media.

Natural language understanding

This converts chunks of text into more formal representations, such as first-order logic structures. Almost all approaches to this task contain these components: lexicon, syntax analyzer and semantic model.

Information extraction (IE)

This is concerned in general with the extraction of structured information (e.g. semantic information) from text. IE includes such tasks as named entity recognition, coreference resolution and relationship extraction.

2.8 Conclusion

In this chapter an overview to the field of machine learning was presented by describing its foundational background and reviewing the basics of different approaches to it.

I presented different approaches to the machine learning tasks, including (i) supervised; (ii) unsupervised; and (iii) semi-supervised.

For supervised approach, three well-known methods were discussed: naive Bayes, maximum entropy and decision tree. Concerning unsupervised learning, I presented two types of clustering: partitional and hierarchical. Finally for semi-supervised approach, two commonly used methods were outlined, namely, Self-Training and Co-Training.

In addition, I presented a short introduction to the field of natural language processing and some related tasks. This Chapter provides the necessary machine learning foundations related to the proposed methods, systems and framework contained in the rest of this thesis.

Chapter 3

Coreference Resolution

3.1 Introduction

This chapter reviews the foundations and related work concerning the task of coreference resolution and the related machine learning methods.

First, the problem of coreference resolution will be formulated. Then, I will continue the review of existing methods in four categories: (i) Rule-based approaches, where we will consider Hobbs [Hobbs, 1986], CogNIAC [Baldwin, 1996], Mitkov [Mitkov, 1998] and Multi-Pass-Sieve [Raghunathan et al., 2010] methods; (ii) Supervised approaches, where a decision tree [Soon et al., 2001], a generative probabilistic [Ge et al., 1998] and a ranking approach [Denis and Baldrige, 2007a] will be discussed; (iii) Unsupervised approaches, where a nonparametric Bayesian model [Haghighi and Klein, 2007] and a Markov logic model [Poon and Domingos, 2008] will be presented; and (iv) Semi-supervised approaches, where a Self-Training [Ng and Cardie, 2003] and a Co-Training [Müller et al., 2002] method for coreference resolution will be considered.

This chapter also presents the most commonly used evaluation metrics of coreference resolution. In order to understand how these metrics work, I present a running example for each metric. I will also discuss the problems associated with each.

This chapter is organized as follows. Section 3.2 formulates the problem of coreference resolution. Four rule-based methods of coreference resolution are described in Section 3.3. Sections 3.4, 3.5 and 3.6 present some related methods

of supervised, unsupervised and semi-supervised approaches. Section 3.7 presents the most commonly used evaluation metrics. In Section 3.8 the chapter will be concluded.

3.2 Problem Formulation

As described in Chapter 2, coreference resolution is the process of finding markables (noun phrases) referring to the same real world entity or concept. In other words, this process groups the markables of a document into equivalence classes (coreference entities), so that all markables in an entity are coreferent.

For example in the following text,

Example 1: *The sales drop for the No. 1 car maker may have been caused in part by the end in September of dealer incentives that GM offered in addition to consumer rebates and low - interest financing , a company spokesman said . Last year , GM had a different program in place that continued rewarding dealers until all the 1989 models had been sold .*

The coreferent markables are underlined: “the No. 1 car maker”, “GM”, - “company” and “GM”. All of these markables refer to the same *organization*, i.e. they form an entity [Hirschman and Chinchor, 1997].

And in the following text,

Example 2: *A few days ago, another Taiwanese star passed away. I had never heard of her. I hear she was a talented beauty.*

All underlined markables refer to the same *person* and form an entity.

Deemter and Kibble [1995] defined coreference more precisely: assume that a_1 and a_2 are occurrences of noun phrases (NPs) and both have a unique referent in the context in which they occur. Taking the functional notation $\text{Referent}(a)$ to mean “the entity referred to by a ”, we can define the coreference relation as follows:

a_1 and a_2 corefer if and only if $\text{Referent}(a_1) = \text{Referent}(a_2)$.

Taking the above definition into account, we can add that coreference is an equivalence relation. In other words, it is reflexive, symmetric and transitive:

Reflexive:

$$\text{Referent}(a_1) = \text{Referent}(a_1)$$

Symmetric:

$$\text{if } \text{Referent}(a_1) = \text{Referent}(a_2) \text{ then } \text{Referent}(a_2) = \text{Referent}(a_1)$$

Transitive:

$$\begin{aligned} &\text{if } \text{Referent}(a_1) = \text{Referent}(a_2) \text{ and } \text{Referent}(a_2) = \text{Referent}(a_3) \\ &\text{then } \text{Referent}(a_1) = \text{Referent}(a_3) \end{aligned}$$

As mentioned earlier, a specific example of *coreference* is the *anaphoric* relation, which has been defined by Deemter and Kibble [1995] as follows: “an NP a_1 is said to take an NP a_2 as its anaphoric antecedent if and only if a_1 depends on a_2 for its interpretation (e.g. [Kamp and Reyle, 1993])”.

Even though coreference and anaphora are not exactly the same concept (i.e. an anaphoric relation is a special case of a coreference relation), some people use the concept of anaphora instead of coreference and vice versa.

There are many coreference related linguistic phenomena. Some of these are [Hoste, 2005]:

1. Nominal anaphora

(a) Names and named entities

- i. Proper names: a noun phrase representing a unique thing (e.g. Berlin, Sun, Obama or Toyota)
- ii. Common names: a noun phrase that describes a class of entities (e.g. city, planet, person or corporation)
- iii. Named entities: a noun phrase that clearly identifies one item from a set of other items with similar attributes (Examples of named entities are first and last names, geographic locations, companies and addresses).

(b) Pronouns:

- i. Personal Pronouns: stand in place of the names of things or people (e.g. she).

- ii. Demonstrative Pronouns: used to distinguish the particular objects or people that are referred to from other possible candidates (e.g. these).
 - iii. Possessive Pronouns: used to indicate ownership or possession (e.g. mine).
 - iv. Indefinite Pronouns: refer to general categories of people or things (e.g. anyone).
 - v. Reflexive Pronouns: used when a person or thing acts on itself (e.g. herself).
- (c) Conjoined noun phrases: a noun phrase contains some other noun phrases (e.g. “all the cars in the world, even a Rolls Royce”).
 - (d) Noun phrases containing relative noun phrases: a relative noun phrase (i.e. a noun phrase in the subordinate clause) is coreferential with the complex noun phrase (e.g. “the boy who ran”).

2. Special cases

- (a) Bound anaphora: coreference between a bound anaphora and the NP which binds it (e.g. “Nobody likes to lose his job.”)
- (b) Appositions: a grammatical construction of two noun phrases placed side-by-side, where one noun phrase defines or modifies the other noun phrase (For example, in the phrase “my friend Hana”, the name “Hana” is in apposition to “my friend”).
- (c) Metonymy: in which a thing or concept is called by the name of something intimately associated with it, instead of its own name. For example, “Westminster” is used as a metonymy for the Parliament of the United Kingdom, because it is located there.

3.2.1 Methods

There are different approaches to the problem of coreference resolution. Roughly speaking we can categorize these approaches into the following groups:

Rule-based Models: usually the rules rely on such features as (i) string matching, (ii) scope (reflexive pronoun, genderless/gendered pronoun), (iii) number, person, gender and Animacy agreement.

Supervised Models: i.e. machine learning approaches which learn by looking at the labeled instances.

Unsupervised Models: i.e. machine learning approaches using unlabeled instances.

Semi-supervised Models: i.e. machine learning approaches that use both labeled and unlabeled instances.

3.2.2 Feature Types

There are different types of features that can be used in machine learning approaches to coreference resolution.

A useful categorization that leads a better overview of the possible feature spaces, distinguished between atomic and pair features, is presented as follows:

Atomic Features: An atomic feature is an attribute. An atomic feature can be defined for a token (word), a markable, a sentence, a paragraph or a document. It is an attribute that is assigned to an object as a unit. The following are examples of atomic features for tokens which can be extracted in preprocessing: Part of speech tag, Grammatical Gender (male, female or neutral), Natural Gender (male or female), Number (e.g. singular, plural or both), Semantic Class, Type (e.g. pronoun types: personal, reflexive, demonstrative ...), Case (e.g. nominative, accusative, dative or genitive in German) and Pronoun Person (first, second or third). Example of atomic features of markables are: number of words in markable, named entity, syntactic role and semantic class. For sentences, the following could be extracted: number of words in the sentence and sentence type (e.g. simple, compound or complex). For paragraphs these features are possible: number of words and number of sentences in the paragraph. Finally, examples of document features include document type (e.g. news, article or book), number of words, sentences and paragraphs in the document.

Pair Features: Pair features are defined over a pair of two objects (two units).

In the context of coreference resolution, pair features concern the features which are defined over a pair of two markables.

3.3 Rule-based Models

There are many rule-based approaches to coreference resolution. Most of the earlier work on coreference resolution focuses on pronominal resolution [Feldman and Sanger, 2007]. Some approaches need heavy preprocessing, whereas some others rely on shallow preprocessing. The following four system descriptions serve to show how a rule-based approach can be taken in different ways.

Criteria for selecting these rule-based systems are the following: *Hobbs* system is considered one of the pioneers in anaphora resolution. *CogNIAC* and *Mitkov* belong to early work on coreference resolution, and they can be considered the classical rule-based models for coreference resolution. *Multi-Pass-Sieve* is a recently introduced system that has the best performance when compared to many other systems.

Although the methods which we have developed in our framework are categorized under statistical machine learning methods, there are two reasons to review the rule-based methods: (i) many features which have been used in these systems are also used in the statistical methods, and (ii) their rules help us to understand many basic ideas which are used in the statistical methods.

3.3.1 Hobbs

It is a simple algorithm for pronominal resolution [Hobbs, 1986]. It works by specifying a total order of the noun phrases in the prior discourse. A parsed representation of the sentences is required. In detail it works as follows:

1. Begin at noun phrase.
2. Go up parse tree to the first noun phrase or sentence node. Call this X , and the path p .
3. Traverse all branches below X to the left of p . Propose as antecedent any noun phrase that has a noun phrase or sentence node between it and X .

4. If X is the tree-root, traverse the parse trees of the previous sentences in the order of recency. Traverse left-to-right, breadth-first. When a noun phrase is encountered, propose as antecedent. If it is not the highest node, go to the next step.
5. From node X , go up the tree to the first noun phrase or S . Call it X , and the path p .
6. If X is noun phrase and the path to X did not pass through the nominal that X dominates, propose X as antecedent.
7. Traverse all branches below X to the right of the path, in a left-to-right, breadth-first manner. Propose any noun phrase encountered as the antecedent.
8. If X is a sentence node, traverse all branches of X to the right of the path but do not go below any noun phrase or sentence node encountered. Propose any noun phrase as the antecedent.

One major criticism of this method is that it is strictly dependent on parse information. Another criticism is that it ignores many coreference cases. Its major limitation, however, is that it is only for pronominal resolution.

The Hobbs [1986] method has an attractive approach to syntactical features which we have used in our supervised system SUCRE (see 4.3). We defined a class of features in which the path between two markables is extracted from the parse tree, looking to see if this path matches the already learned patterns.

3.3.2 CogNIAC

CogNIAC is also a pronoun resolution engine. It assumes that there is no need for general purpose reasoning for all cases of anaphora. It is based on a set of high confidence rules [Baldwin, 1996]. The six rules used by system are listed below [Feldman and Sanger, 2007]. Sentence prefix of anaphora means the text portion beginning from the first of sentence to the position of the anaphora.

Unique Antecedent:

Condition: there is a single compatible antecedent A in the corresponding

discourse.

Action: A is selected as the antecedent.

Reflexive Pronoun:

Condition: the anaphor is a reflexive pronoun.

Action: take the nearest compatible antecedent in the current sentence prefix of anaphor.

Unique in Current and Preceding:

Condition: there is a single compatible antecedent A in the preceding sentence and the current sentence prefix of anaphor.

Action: A is selected as the antecedent.

Possessive Pronoun:

Condition: the anaphor is a possessive pronoun and there is an exact copy of the possessive phrase in the previous sentence.

Action: the antecedent of the copy is the same as the former.

Unique in Current:

Condition: there is a single compatible antecedent A in the current sentence prefix of anaphor.

Action: A is selected as the antecedent.

Unique Subject-Subject Pronoun:

Condition: the subject of the previous sentence is a compatible antecedent A and the anaphor is the subject of the current sentence.

Action: A is selected as the antecedent.

CogNIAC is a high precision pronoun resolution system. However, such approaches tend to ignore many coreference cases. A major problem of this approach is that it works only for pronominal resolution.

The way in which the *CogNIAC* system selects the antecedent in “Unique Antecedent”, “Unique in Current and Preceding” and “Unique in Current” rules is very interesting and has motivated the development of the chain estimation algorithm that will be presented in section 4.5. A clustering method for chain estimation was developed that searches for the best predicted antecedent above a threshold from right-to-left, starting from the end of the document.

3.3.3 Mitkov

This pronoun resolution method is based on a set of boosting and impeding indicators. It uses a part-of-speech tagger to identify the noun phrases which precede the anaphor within a distance of 2 sentences. It checks then the gender and number agreement between the noun phrases and the anaphor. Finally, it applies genre-specific antecedent indicators to the remaining noun phrases that could pass the last steps, taking the noun phrase with the highest aggregate score as antecedent. The antecedent indicators used by Mitkov [1998] are listed below:

Definiteness

Compared to indefinite noun phrases, the definite noun phrases in previous sentences are more likely to be antecedents of pronominal anaphora. Therefore definite noun phrases are scored as 0 and indefinite ones are penalized as -1.

Givenness

Noun phrases in previous sentences representing the “given information” are all good candidates to be antecedent. Due to this assumption, such noun phrases are scored as 1, and the other noun phrase are scored as 0. Mitkov used the simple heuristics to determine the given information. It is the first noun phrase in a non-imperative sentence.

Indicating verbs

If a verb in the sentence has one of these stems: discuss, present, illustrate, identify, summarize, examine, describe, define, show, check, develop, review, report, outline, consider, investigate, explore, assess, analyze, synthesize, study, survey, deal or cover, then the first NP following it is taken as the preferred antecedent and is scored as 1, and the other noun phrase as 0.

Lexical reiteration

The noun phrases that are reiterated are more likely to be antecedent. Therefore, a noun phrase is scored as 2 if it is repeated within the same paragraph twice or more, is scored as 1 if repeated once, and as 0 if it is not repeated.

Section heading preference

A noun phrase that is in the heading of the section has more chance of being antecedent. Therefore it is scored as 1; the rest as 0.

“Non-prepositional” noun phrases

In comparison to a “pure”, “non-prepositional” noun phrase, a prepositional phrase is less likely to be corefered. Then the prepositional phrases are scored as -1, otherwise as 0.

Collocation pattern preference

A candidate with the identical collocation pattern is given preference (is scored as 2). The applied collocation patterns are: “noun phrase verb” and “verb noun phrase”.

Immediate reference

The “immediate reference” is a useful clue for identifying the antecedent in technical manuals. It can be recognized by a heuristic method. For example given a pattern of the form “...You? V1 NP ...CN you? V2 it (CN you? V3 it)”, where $CN \in \{and, or, before, after \dots\}$, the noun phrase immediately after V1 is a very likely coreference candidate for the pronoun “it” immediately following V2 and therefore is scored as 2.

Referential distance

In complex sentences, distance between the candidate antecedent and the anaphor is a good measure to look at. Therefore noun phrases receive the following scores based on their distance to the anaphor:

1. Previous clause: 2
2. Previous sentence: 1
3. Two sentences: 0

And in simple sentences, the scores are:

1. Previous sentence: 1
2. Two sentences: 0
3. Three sentences and more: -1

Term preference

In comparison with the noun phrases which are not terms, the noun phrases that represent domain terms are more likely to be the antecedent. Therefore if the noun phrase is a term then it is scored as 1, and 0 if not.

While Mitkov’s work improves on prior work, it has been criticized because it ignores many coreference cases and works only for pronominal resolution.

An interesting feature that has been used by Mitkov [1998] is “Lexical reiteration”. The approach that we took for development of the unsupervised system A-INF has been motivated by this early work. We will present A-INF in section 6.4.

3.3.4 Multi-Pass-Sieve

In contrast to the previous approaches that perform pronoun resolution, Multi-Pass-Sieve is a full coreference resolution system [Raghunathan et al., 2010]. It proposes a simple coreference architecture based on a sieve comprising some tiers of deterministic coreference models. It applies the tiers one at a time, from highest to lowest precision. Each tier builds on the previous tier entity cluster output. The tiers (passes) are as follows:

Pass 1 - Exact Match

This is the most precise model, with a pairwise precision over 96%. It is applied first. It links two markables only if they contain exactly the same extent text, including modifiers and determiners.

Pass 2 - Precise Constructs

This model links two markables if any of the following conditions are satisfied:

1. **Appositive:** the two nominal noun phrases are in an appositive construction, e.g., “*US President*, Barack Obama, said ...”
2. **Predicate nominative:** there is a copulative subject-object relation between the two markables (nominal or pronominal). e.g., *The New York-based College Board* is a nonprofit organization [Poon and Domingos, 2008].

3. **Role appositive:** the candidate antecedent, whose head is a noun, appears as a modifier in a noun phrase whose head is the current markable, e.g., *actress Rebecca Schaeffer*.
4. **Relative pronoun:** the markable is a relative pronoun that modifies the head of the antecedent noun phrase, e.g., *the finance street which has already formed in the Waitan district*.
5. **Acronym:** a markable is an acronym of the other. A mention is an acronym of another if its text matches the sequence of upper case characters in the other mention. e.g., *Agency France Presse ... AFP*.
6. **Demonym:** a markable is a demonym of the other. A demonym is a name for a resident of a place. It is derived from the name of the particular place. e.g., *Germany ... German*.

Pass 3 - Strict Head Matching

Linking of two markables based on the naive head word matching can generate a lot of spurious links. This is because possibly incompatible modifiers could be ignored [Elsner and Charniak, 2010]. For example, “Stanford University” and “Stuttgart University” have similar head words, but they are not the same entity. The following features address this issue (they must all be matched in order to yield a link):

1. **Cluster head match:** the markable head word matches any head word in the antecedent cluster. This feature is more relaxed than naive head matching but is constrained by enforcing a conjunction with the features below.
2. **Word inclusion:** all the non-stop words in the cluster of markable are included in the set of non-stop words in the cluster of the antecedent candidate.
3. **Compatible modifiers only:** the markable’s modifiers are all included in the modifiers of the antecedent candidate.
4. **Not i-within-i:** the two markables are not in an i-within-i construct, i.e., it cannot be a child noun phrase in the constituent of other noun phrase [Haghighi and Klein, 2009].

Pass 4 - Variant of Strict Head Matching

This pass is a variant of strict head matching by removing the “compatible modifiers only” feature.

Pass 5 - Variant of Strict Head Matching

This pass is a variant of strict head matching in which the word inclusion constraint is removed.

Pass 6 - Relaxed Head Matching

This pass is a relaxation of the cluster head matching by allowing markable head matches to any word in the cluster of the candidate antecedent.

Pass 7 - Pronouns

This pass focuses on pronominal coreference resolution that except for the second pass all previous passes ignore. Similar to previous work, pronominal coreference resolution is implemented by enforcing agreement constraints between the markables. The following features are used for this purpose:

1. **Number:** this feature is assigned based on (i) a static list for pronouns, (ii) named entity labels, (iii) part of speech tags, and (iv) a static dictionary from [Bergsma and Lin, 2006].
2. **Gender:** this is assigned based on static lexicons from Bergsma and Lin [2006] and Ji and Lin [2009].
3. **Person:** this feature is assigned only to pronouns.
4. **Animacy:** this feature is extracted from: (i) a static list for pronouns; (ii) named entity labels, e.g., PERSON is animate whereas ORGANIZATION is not; and (iii) a dictionary bootstrapped from the web [Ji and Lin, 2009].
5. **Named Entity:** if no value is detected, the feature is set to unknown and is treated as a wild card, i.e., they can match any other value.

Multi-Pass-Sieve is a complex rule-based system that performs better than all other rule-based and learning-based approaches. Nevertheless, it has not escaped criticism. A limitation of complex rule-based systems is that they require substantial effort to encode the large number of deterministic constraints that

guarantee good performance. Moreover, these systems are not adaptable (since they are not machine-learned) and may have to be rewritten for each new domain, genre and language.

3.4 Supervised Models

The supervised approaches use manually labeled data. Ng [2010] divides the supervised approaches into three categories:

1. Markable-Pair Model: there is a classifier that determines whether two markables are coreferent (e.g. [Soon et al., 2001]).
2. Cluster-Markable Model: aims to classify whether a markable is coreferent with a preceding cluster (e.g. [Ge et al., 1998]).
3. Ranking Models: try to determine which candidate antecedent is most probable given a markable to be resolved (e.g. [Denis and Baldrige, 2007a]).

The following systems are examples of supervised approaches to coreference resolution for each of the above mentioned categories.

The supervised subsystem of our framework, which will be presented in chapter 4, is a Markable-Pair model. Compared with other supervised models, my model provides a more flexible method for feature engineering that supports rapid feature definition and extraction.

3.4.1 Decision Tree

Soon et al. [2001] presented a learning approach to coreference resolution of noun phrases in unrestricted text using a decision tree. Their approach can be used for nominal and pronominal noun phrases.

They took a classical markable-pair approach to coreference resolution that tackles the problem in two steps (e.g. [Tetreault, 2001], [Ng and Cardie, 2002]):

1. **Classification:** coreference is determined between markable pairs (links). In other words, the markable pairs are classified as *coreferent* or *disreferent*.
2. **Partitioning:** coreferent markable pairs are used as basic building blocks to arrive at a partition.

This approach is very similar to clustering, where the aim is to partition a set of objects into clusters on the basis of similarity judgments on pairs of objects [Wagstaff, 2002].

First, all possible markables in a training document are detected by a pipeline of language-processing modules. Then the training instances, in the form of feature vectors, are generated for the appropriate markable pairs. These training instances are then given to a learning algorithm that can build a decision tree. For a new document, in order to determine the coreference chains (partitioning), all markables are detected and potential coreferring markable pairs are given to the classifier that decides whether the two markables are coreferent or disreferent (classification).

The applied feature vector consists of a total of 12 features which are described below. It is calculated over two candidate markables, i and j , where i is the potential antecedent, and j is the anaphor.

1. **Distance Feature (DIST):**

This feature captures the sentence distance between i and j . Its possible values are 0,1,2,3,...

2. **i-Pronoun Feature (I_PRONOUN):**

This feature returns true if i is a pronoun; otherwise it returns false. Pronouns include reflexive pronouns (himself, herself), personal pronouns (he, him, you), and possessive pronouns (hers, her).

3. **j-Pronoun Feature (J_PRONOUN):**

This feature returns true if j is a pronoun; otherwise it returns false.

4. **String Match Feature (STR_MATCH):**

It returns true if the string of i matches the string of j ; otherwise it returns false. The articles (a, an, the) and demonstrative pronouns (this, these, that, those) are removed from the strings before comparing the strings (e.g. “the license” matches “this license”).

5. **Definite Noun Phrase Feature (DEF_NP):**

If j is a definite noun phrase, it returns true; otherwise it returns false. A definite noun phrase is defined as a noun phrase that starts with the word “the” (e.g. the car is a definite noun phrase).

6. Demonstrative Noun Phrase Feature (DEM_NP):

If j is a demonstrative noun phrase, it returns true; otherwise it returns false. A demonstrative noun phrase is defined as a noun phrase that starts with the word “this”, “that”, “these”, or “those” (e.g. “this car” is a demonstrative noun phrase).

7. Number Agreement Feature (NUMBER):

If i and j agree in number (i.e., both are singular or both are plural), this feature is true; otherwise it is false.

8. Semantic Class Agreement Feature (SEMCLASS):

If i and j agree in semantic class, this feature is true; otherwise it is false. Soon et al. defined the following semantic classes: “female, male, person, organization, location, date, time, money, percent and object”.

9. Gender Agreement Feature (GENDER):

If i and j agree in gender, this feature is true; otherwise it is false. The gender of a markable is determined in several ways.

10. Both-Proper-Names Feature (PROPER_NAME):

If i and j are both proper names, this feature is true; otherwise it is false. A proper name can be determined based on capitalization.

11. Alias Feature (ALIAS):

If i is an alias of j or vice versa, it returns true; otherwise it returns false.

12. Appositive Feature (APPOSITIVE):

If j is apposition of i , it returns true; otherwise it returns false.

One major drawback of this approach is that it cannot determine how good a candidate antecedent is relative to other candidates [Rahman and Ng, 2011].

The supervised subsystem of the framework which I will present in chapter 4 also uses Decision Tree. However, SUCRE has a more comprehensive feature set and provides a more flexible method for feature engineering that supports rapid feature definition and extraction.

3.4.2 Generative Probabilistic Approach

This system utilizes a generative probabilistic approach to anaphora resolution [Ge et al., 1998], which can be categorized under cluster-markable models. It is a statistical method for determining pronoun anaphora by introducing the following equation, where $F(\rho)$ denotes a function from pronouns to their antecedents (cluster of candidate antecedents):

$$F(\rho) = \operatorname{argmax}_a P(A(\rho) = a | \rho, h, \vec{W}, t, l, s_p, \vec{d}, \vec{M}) \quad (3.1)$$

where $A(\rho)$ represents a random variable indicating the referent of the pronoun ρ , and a is a proposed antecedent. In the conditioning events of the same equation:

- h is the head constituent above ρ .
- \vec{W} is the list of candidate antecedents to be taken in account.
- t is the phrase type of the proposed antecedent.
- l is the type of the head constituent.
- s_p describes the syntactic structure in which ρ appears.
- \vec{d} contains the distance of each antecedent from ρ .
- \vec{M} is the number of times the referent is mentioned.

\vec{W} , \vec{d} and \vec{M} are vector quantities that contain the corresponding entries of candidate antecedents. With this approach to the problem, a can be seen as an index into these vectors. By applying Bayes' theorem:

$$P(A(\rho) = a | \rho, h, \vec{W}, t, l, s_p, \vec{d}, \vec{M}) = \frac{P(a | \vec{M}) \times P(\rho, h, \vec{W}, t, l, s_p, \vec{d} | a, \vec{M})}{P(\rho, h, \vec{W}, t, l, s_p, \vec{d} | \vec{M})} \quad (3.2)$$

From the above equation, it is possible to arrive at the following equation:

$$\propto P(a | \vec{M}) \times P(d_H | a) \times P(\vec{W} | h, t, l, a) \times P(\rho | w_a) \quad (3.3)$$

where the syntactic structure s_p , and the distance from the pronoun \vec{d} are independent of the number of times the referent is mentioned and can be covered by one variable d_H (a.k.a. Hobbs distance).

And finally from equation 3.3, the following equation can be concluded (for more details, see [Ge et al., 1998]):

$$P(A(\rho) = w_a) \propto P(d_H|a) \times P(\rho|w_a) \times \frac{P(w_a|h, t, l)}{P(w_a|t, l)} \times P(a|m_a) \quad (3.4)$$

In equation 3.4, $P(d_H|a)$ can be obtained by running the Hobbs algorithm on the training data [Hobbs, 1986]. The probability $P(\rho|w_a)$ is easily obtained by using the training corpus, which is tagged with reference information. $P(w_a|h, t, l)$ and $P(w_a|t, l)$ are available from previous work [Charniak, 1997a]. The corpus also contains referents' repetition information, which gives $P(a|m_a)$. Equation 3.4 has four components which can be estimated in a reasonable fashion. By computing the product of these components, the system can return the antecedent w_a for a pronoun ρ that maximizes the corresponding probability.

The main weakness of this method is that it is only for pronominal resolution. However, cluster-markable models are more powerful than markable-pair models, at least in principle. While we are planning to look at cluster-markable models in future work, but the focus in this thesis is on markable-pair models because they are more efficient and parameters can be estimated more reliably.

3.4.3 Ranking Approach

Denis and Baldrige [2007a] proposed a supervised maximum entropy ranking approach to pronoun resolution as an alternative to the commonly used classification-based approaches. In the classification approaches only one or two candidate antecedents are considered for a pronoun at a time, whereas ranking considers all candidates to be evaluated at the same time.

Before looking at the ranking approach, we need to take a look at the formal presentation of two other approaches: first; the single-candidate classification approach to pronoun resolution (similar to the approaches Soon et al. [2001], or Culotta et al. [2007]); and second, the twin-candidate model that was proposed by Yang et al. [2003a].

A single-candidate approach tackles the problem in two steps by:

1. estimating the probability, $P_c(COREF|\langle\pi, a_i\rangle)$, of having a coreferential

outcome given a pair of markables $\langle \pi, a_i \rangle$, where π is an anaphoric pronoun, and $A = a_1, \dots, a_n$ is a set of antecedent candidates.

2. applying a selection algorithm that will single a unique candidate out of the subset of candidates a_k for which the probability $P_c(COREF|\langle \pi, a_k \rangle)$ is above a threshold (typically .5).

In this case, for a given pronoun, the number of events is just the cardinality of the set of candidates.

$$P_c(COREF|\langle \pi, a_i \rangle) = \frac{\exp(\sum_{i=1}^n \lambda_i f_i(\langle \pi, a_i \rangle, COREF))}{\sum_c \exp(\sum_{i=1}^n \lambda_i f_i(\langle \pi, a_i \rangle, c))} \quad (3.5)$$

For training, the instances are generated based on pairs of markables of the form $\langle \pi, a_i \rangle$. Depending on whether or not the two markables are coreferent, they are assigned either a label COREF (i.e. a positive instance) or a label NOT-COREF (i.e. a negative instance). Then for each anaphoric pronoun, the system generates: **(i)** a positive instance for the pair $\langle \pi, a_i \rangle$, where a_i is the closest antecedent for π , and **(ii)** a negative instance for each pair $\langle \pi, a_j \rangle$, where a_j intervenes between a_i and π .

A twin-candidate approach is also a two step process:

1. estimating the probability, $P_c(FIRST|\langle \pi, a_i, a_j \rangle)$, of the pronoun π being coreferent with the first antecedent candidate a_i . It is a binary classification with the dual probability $P_c(SECOND|\langle \pi, a_i, a_j \rangle)$, of the pronoun π being coreferent with the second antecedent candidate a_j .
2. like the single-candidate approach, there is also a selection algorithm. Here, the selection algorithm involves comparing candidates in pairs.

$$P_{tc}(FIRST|\langle \pi, a_i, a_j \rangle) = \frac{\exp(\sum_{i=1}^n \lambda_i f_i(\langle \pi, a_i, a_j \rangle, FIRST))}{\sum_c \exp(\sum_{i=1}^n \lambda_i f_i(\langle \pi, a_i, a_j \rangle, c))} \quad (3.6)$$

For training, the instances are generated based on triples of markables of the form $\langle \pi, a_i, a_j \rangle$, where π is a pronominal anaphor. a_i and a_j are the two of its candidate antecedents. a_i is closer to π than a_j . If a_i is the correct antecedent, then the instance is labeled with FIRST, and if a_j is the correct antecedent, then the instance is labeled with SECOND.

Now the ranking approach which was introduced by Denis and Baldridge [2007a] will be presented. In the ranking approach, pronoun resolution is done in a single step. It computes the probability $P_r(a_i|\pi)$, of a particular candidate a_i being coreferent with the anaphoric pronoun π . A unique event is created for each pronoun and its entire candidate set A . And in the final stage, the correct antecedent is the most likely candidate in this set:

$$P_{tc}(a_i|\pi) = \frac{\exp(\sum_{i=1}^n \lambda_i f_i(\pi, a_i))}{\sum_k \exp(\sum_{i=1}^n \lambda_i f_i(\pi, a_k))} \quad (3.7)$$

The training instances for the ranker system are generated based on an anaphoric pronoun π and its antecedent candidates in a window of 4 sentences around it.

In principle, ranking models – just as cluster-markable models are – are more powerful than markable-pair models. We focus on markable-pair models in this dissertation because they are more efficient and parameters can be estimated more reliably.

3.5 Unsupervised Models

The unsupervised learning approaches do not use manually-labeled data. Therefore their performance is usually lower than supervised approaches. The following unsupervised systems do however, achieve performance close to the supervised ones. For this reason, we present them in detail.

3.5.1 Nonparametric Bayesian Model

Haghighi and Klein [2007] proposed a nonparametric Bayesian approach to unsupervised coreference resolution. Their model is fully generative, producing each mention from a combination of global entity properties and local attentional state.

Taking a collection of I documents, each with J_i mentions, and a random variable Z to refer to (indices of) entities, the goal is to find the setting of the entity indices which maximize the posterior probability:

$$Z^* = \arg \max_Z P(Z|X) = \arg \max_Z P(Z, X) = \arg \max_Z \int P(Z, X, \phi) dP(\phi) \quad (3.8)$$

In the above equation, ϕ refers to the concatenation of all the parameters for an entity z . And X refers somewhat loosely to the collection of variables associated with a mention in the model (such as the head or gender). For performing the inference, a Gibbs sampling procedure was developed to obtain samples from $P(Z|X)$. (For more details, see [Haghighi and Klein, 2007])

A serious weakness with this model, however, is its sampling approach (cluster size and cluster number grow logarithmically with the number of data points).

In chapter 6, my proposed framework will be directly compared to Haghighi and Klein [2007] system.

3.5.2 Markov Logic Model

Poon and Domingos [2008] presented a joint unsupervised coreference resolution model with Markov logic. It is the first unsupervised approach that can compete with supervised ones. Their model performs joint inference across markables, in contrast to the pairwise approach. They used the Markov logic as a representation language to easily handle relations like apposition and predicate nominal. A set of weighted first-order clauses with a set of constants define a Markov logic network (MLN). MLN is a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of its original first-order clause. The probability of a state x in such a network is given by

$$P(x) = \frac{\exp(\sum_i w_i f_i(x))}{Z} \quad (3.9)$$

where Z is a normalization constant, and w_i is the weight of clause i . If clause i is true, then $f_i = 1$. If false, $f_i = 0$.

They used a slice sampling Markov chain Monte Carlo algorithm for inference. Slice sampling employs auxiliary variables u that decouple the original ones x , and it alternately samples u conditioned on x , and vice versa.

One question that needs to be asked, however, is whether it allows all candidates to be considered simultaneously.

We will directly compare our framework with Poon and Domingos [2008] system in chapter 6.

3.6 Semi-supervised Models

As described before, semi-supervised learning approaches use both labeled and unlabeled data for training (typically a small amount of labeled data with a large amount of unlabeled data). Below we will look at existing systems, which are related to each of two classical semi-supervised approaches that we discussed in Section 2.6.

3.6.1 Self-Training (bootstrapping)

Ng and Cardie [2003] proposed a self-training approach by bootstrapping coreference classifiers with two machine learning algorithms. They used naive Bayes (see Equation 2.9) and decision list as the learners of their multiple-learner framework. As both classifiers have the same view on the data (i.e. same feature set), we consider this approach a self-training approach.

A decision list is a simple learning algorithm that consists of a list of decision rules, where a rule is assigned for one possible value v_j of a feature f_i in the training data [Collins and Singer, 1999]. The rules in the list are sorted in decreasing order of their decision power, which is defined as the conditional probability $P(y|f_i = v_j)$ and is estimated using the training data as follows:

$$P(y|f_i = v_j) = \frac{N(f_i = v_j, y) + \alpha}{N(f_i = v_j) + k\alpha}$$

where $N(x)$ is the frequency of event x in the training data; α , a smoothing parameter; and k , the number of classes.

Taking the above mentioned classifiers, at each bootstrapping iteration all instances of the data pool are scored by each classifier. The instances which have the highest scores from one classifier are added to the training data of the other classifier and vice versa. The two classifiers are trained on the same feature set. The training set of each classifier is separately maintained.

Similar to the algorithm presented in Figures 2.3, the above algorithm can be formulated as in Figure 3.1.

However, such methods tend to oversimplify the problem when doing the bootstrap analysis, because the seed labeled data is usually a small dataset. In other words, it is possible that some existing characteristics are not recognizable

procedure *TwoClassifierSelfTrain*(L_0, U)

1. L_0 is seed labeled data
2. L_{DL} is labeled data by Decision List classifier
3. L_{NB} is labeled data by Naive Bayes classifier
4. U is unlabeled data
5. $c_{DL} \leftarrow \text{train}_{DL}(L_0)$
6. $c_{NB} \leftarrow \text{train}_{NB}(L_0)$
7. $L_{DL} \leftarrow L_0 + \text{select}(\text{label}(U, c_{NB}))$
8. $L_{NB} \leftarrow L_0 + \text{select}(\text{label}(U, c_{DL}))$
9. $c_{DL} \leftarrow \text{train}_{DL}(L_{DL})$
10. $c_{NB} \leftarrow \text{train}_{NB}(L_{NB})$
11. if no stopping criterion is met, go to step 7
12. return $c_{DL} + c_{NB}$

Figure 3.1: An illustration of the method proposed by Ng and Cardie [2003] for bootstrapping coreference classifiers.

from the seed data samples. The Self-Training approach that I will present in Chapter 6 overcomes this problem by using a very large unlabeled data set as the seed data.

3.6.2 Co-Training

Müller et al. [2002] investigated the practical applicability of Co-Training for reference resolution. They aimed to answer the question as to whether Co-Training can significantly reduce the amount of manual labeling work while at the same time producing a classifier with an acceptable performance.

They used a set of 17 features and determined the features for the two different views of Co-Training (see Figure 2.4) as follows: train classifiers on each feature separately and select the best one as the first feature of view 1. Delete the selected feature from the feature set and perform the above step for view 2. Then repeat the above two steps for feature but combine the already selected feature with each remaining feature until all features are either in view 1 or view 2. Figure 3.2 presents this algorithm.

Co-Training is an appropriate method for problems where features clearly derive from different modalities or other sources that are fundamentally different. Müller et al. [2002] fail to show that this is true of coreference resolution. In fact, the induced separation of features into two classes can be seen as an ad-hoc induction of modalities that may not be a good approach to the problem.

3.7 Evaluation Metrics

Evaluation is an important step of system development and testing. An adequate evaluation metric should fulfill the following requirements [Luo, 2005]:

Discriminativity: the ability to differentiate a better performing systems from another one. This is a basic requirement that every metric must have. Although it sounds trivial, many metrics fail to provide it.

Interoperability: the degree of interpretation ability. A good metric should be able to tell us about the quality and quantity of a system output. In other words, it should provide an intuitive sense how good a system performs. In

procedure *CoTrainFeatures*(F)

1. F is the set of all features
2. F_1 is the set of features of view 1 (empty initialized).
3. F_2 is the set of features of view 2 (empty initialized).
4. $f_t \leftarrow \arg \max_{f_i \in F} \text{train}(F_1 \cup \{f_i\})$
5. $F_1 \leftarrow F_1 \cup \{f_t\}$
6. $F \leftarrow F - \{f_t\}$
7. $f_t \leftarrow \arg \max_{f_i \in F} \text{train}(F_2 \cup \{f_i\})$
8. $F_2 \leftarrow F_2 \cup \{f_t\}$
9. $F \leftarrow F - \{f_t\}$
10. if there is any feature in F , go to step 4
11. return F_1 and F_2

Figure 3.2: Feature selection for Co-Training.

addition, it should be possible to judge a system without comparing with other systems.

There is a long debate about the evaluation metrics which can be used for coreference resolution. For several years, the MUC metric was the main standard [Vilain et al., 1995]. MUC metric has a few serious shortcomings. B³ [Bagga and Baldwin, 1998] was introduced later to overcome the shortcomings of MUC metric. However it has by itself some other problems. There are some other evaluation metrics, namely: $CEAF^M$, $CEAF^E$ [Luo, 2005] and recently introduced BLANC . They aimed to solved the problems of MUC and B³, but they have not been completely successful and suffer from their own shortcomings.

In the following we study the details of above mentioned metrics.

3.7.1 MUC

MUC is the first and most widely used evaluation metric of the coreference resolution task. It is the official scorer of the MUC-6 [Chinchor and Sundheim, 2003] and MUC-7 [Hirschman and Chinchor, 1997] tasks on coreference resolution, which have been developed by Vilain et al. [1995].

In order to understand how the scoring metric works, we need to define the terms *key* and *response*. *key* refers to the manually annotated coreference chains (gold chains), and *response* to the coreference chains generated by a system [Baldwin et al., 1998]. We define S as an equivalence set generated by the key, and R_1, \dots, R_m as equivalent classes generated by the response, and the following functions over S as follows:

$p(S)$: a partition whose subsets are generated by intersecting S and the corresponding response sets R_i (i.e. those which overlap S). For example, if $S = \{A, B, C, D\}$ and response is simply equal to $\{A, B\}$, then $p(S) = \{\{A, B\}, \{C\}, \{D\}\}$.

$c(S)$: the minimal number of “correct” links that is needed to build the equivalence class S . It is clear that $c(S) = (|S| - 1)$.

$m(S)$: the number of “missing” links in the response that is needed to fully reunite any components of the $p(S)$ partition. It is clear that $m(S) = (|p(S)| - 1)$.

Focusing on a single equivalence class t in the key, its recall error is calculated as follows

$$Recall_t \text{ error} = \frac{m(S)}{c(S)}$$

And consequently recall is

$$Recall_t = \frac{c(S) - m(S)}{c(S)}$$

The above equation can be rewritten as

$$Recall_t = \frac{|S| - |p(S)|}{|S| - 1}$$

Finally, if we extend the above measure to the entire set (i.e. summing over the key equivalence classes), the recall is

$$Recall = \frac{\sum_i |S_i| - |p(S_i)|}{\sum_i |S_i| - 1}$$

And precision is computed using the above formulation by switching the roles of the key and response. And F_1 measure is computed as follows

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (3.10)$$

MUC has two problems:

1. The identification of singletons (entities with a single markable) is not addressed. This comes from the fact that MUC is based on the number of “missing” links in the response, and a singleton has no missing inside link that causes any penalty.
2. All mistakes a system makes are considered to be equal. For example, a system incorrectly merging 2 entities each with 2 markables receives the same penalty as one incorrectly merging 2 entities each with 20 markables.

Example: by way of illustration, the following example shows how each metric works.

Suppose Figure 3.3 shows the key partition containing 3 chains of a total of 9 markables. We imagine 3 different response partitions. *Response partition 1*,

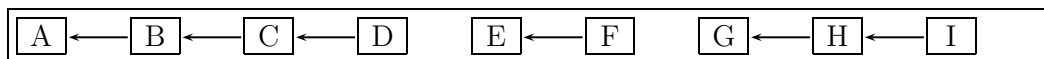


Figure 3.3: Example: key partition containing 3 chains.

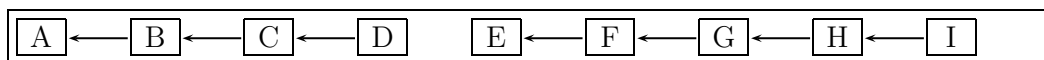


Figure 3.4: the response partition 1 containing 2 chains.

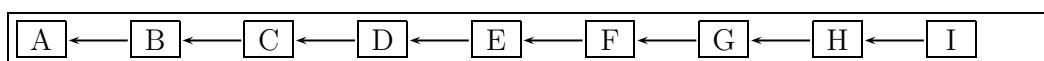


Figure 3.5: the response partition 2 (all markables in one chain).



Figure 3.6: the response partition 3 (each markable in its own chain).

which is illustrated in Figure 3.4, contains 2 chains. *Response partition 2*, which is showed in Figure 3.5, contains 1 chain (all markable in one chain). And finally *response partition 3*, which is presented in Figure 3.6, contains 9 chains (each markable in its own chain).

Response partitions 2 and 3 represent two baseline strategies of coreference resolution:

1. ONE-CLUSTER: all the markables form one entity.
2. SINGLETONS: each markable is an entity.

Then the MUC_{recall} , $MUC_{precision}$ and MUC_{F_1} scores of the response partitions are:

Response Partition 1:

$$MUC_{recall} = \frac{(4 - 1) + (2 - 1) + (3 - 1)}{(4 - 1) + (2 - 1) + (3 - 1)} = 100\%$$

$$MUC_{precision} = \frac{(4 - 1) + (2 - 1) + (3 - 1)}{(4 - 1) + (5 - 1)} = 85.7\%$$

$$MUC_{F_1} = \frac{2 \times 100 \times 85.7}{100 + 85.7} = 92.3\%$$

Response Partition 2:

$$MUC_{recall} = \frac{(4 - 1) + (2 - 1) + (3 - 1)}{(4 - 1) + (2 - 1) + (3 - 1)} = 100\%$$

$$MUC_{precision} = \frac{(9 - 3)}{(9 - 1)} = 75\%$$

$$MUC_{F_1} = \frac{2 \times 100 \times 75}{100 + 75} = 85.7\%$$

Response Partition 3:

$$MUC_{recall} = \frac{0}{0} = -$$

$$MUC_{precision} = \frac{0}{(9 - 1)} = 0\%$$

$$MUC_{F_1} = -$$

3.7.2 B-Cubed

B-Cubed (B^3) was proposed by Bagga and Baldwin [1998]. Instead of looking at the links produced by a system (as MUC does), it looks at the presence/absence of key entities relative to the entities in the produced equivalence classes. It computes the precision and recall numbers for each entity in the document.

To present a model theoretic form of B^3 , the following functions over S are defined:

$p(S)$: like the aforementioned description, it is a partition whose subsets are generated by intersecting S and the corresponding response sets R_i (i.e. those that overlap S). Let $p(S) = \{P_1, P_2, \dots, P_m\}$, where each P_j is a subset of S .

$m_j(S)$: the number of “missing” elements in each P_j relative to the key set S . It is clear that $m_j(S) = (|S| - |P_j|)$.

The number of missing entities for the entire set S is computed as

$$\sum_{j=1}^m \sum_{\text{for each } e \in P_j} m_j(S)$$

Recall for entire set is equal to

$$1 - \frac{\sum_{j=1}^m \sum_{\text{for each } e \in P_j} m_j(S)}{|S|^2}$$

That can be simplified to

$$1 - \frac{\sum_{j=1}^m \sum_{\text{for each } e \in P_j} |S| - |P_j|}{|S|^2}$$

If there are n entities in the documents, the measure can be extended from a single key equivalence class to a set $T = \{S_1, S_2, \dots, S_n\}$ of equivalence classes. Then recall $Recall_i$ of equivalence class S_i is

$$Recall_i = 1 - \frac{\sum_{j=1}^m \sum_{\text{for each } e \in P_{ij}} |S_i| - |P_{ij}|}{|S_i|^2}$$

where P_{ij} is the j th element of the partition $p(S_i)$.

And final recall is computed as follows

$$Final\ Recall = \sum_{i=1}^n \frac{|S_i|}{\sum_{j=1}^n |S_j|} Recall_i$$

As for the MUC score, the precision is calculated by reversing the roles of the key and response.

In contrast to MUC, a shortcoming of B^3 is that it gives too much credit to the identification of singletons. This creates serious problems when the corpus has a high amount of singleton entities.

Example: Using the same representation of the examples presented in Figures 3.3-3.6, the B^3_{recall} , $B^3_{precision}$ and $B^3_{F_1}$ scores are

Response Partition 1:

$$\begin{aligned} B^3_{recall} &= \left(\frac{4}{9} \times \frac{4}{4} + \frac{2}{9} \times \frac{2}{2} + \frac{3}{9} \times \frac{3}{3} \right) = 100\% \\ B^3_{precision} &= \left(\frac{4}{9} \times \frac{4}{4} + \frac{2}{9} \times \frac{2}{5} + \frac{3}{9} \times \frac{3}{5} \right) = 73.3\% \\ B^3_{F_1} &= \frac{2 \times 100 \times 73.3}{100 + 73.3} = 84.6\% \end{aligned}$$

Response Partition 2:

$$\begin{aligned}
 B_{recall}^3 &= \left(\frac{4}{9} \times \frac{4}{4} + \frac{2}{9} \times \frac{2}{2} + \frac{3}{9} \times \frac{3}{3} \right) = 100\% \\
 B_{precision}^3 &= \left(\frac{4}{9} \times \frac{4}{9} + \frac{2}{9} \times \frac{2}{9} + \frac{3}{9} \times \frac{3}{9} \right) = 35.8\% \\
 B_{F_1}^3 &= \frac{2 \times 100 \times 35.8}{100 + 35.8} = 52.7\%
 \end{aligned}$$

Response Partition 3:

$$\begin{aligned}
 B_{recall}^3 &= \left(\frac{4}{9} \times \frac{1}{4} + \frac{2}{9} \times \frac{1}{2} + \frac{3}{9} \times \frac{1}{3} \right) = 33.3\% \\
 B_{precision}^3 &= \left(\frac{4}{9} \times \frac{1}{1} + \frac{2}{9} \times \frac{1}{1} + \frac{3}{9} \times \frac{1}{1} \right) = 100\% \\
 B_{F_1}^3 &= \frac{2 \times 100 \times 33.3}{100 + 33.3} = 50.0\%
 \end{aligned}$$

3.7.3 CEAF

The CEAF metric has two types, namely $CEAF^M$ and $CEAF^E$. It was proposed by Luo [2005].

The key idea behind CEAF metrics is the entity similarity metric ϕ , which maps each response cluster R_i to the most similar key entity S_j .

Two simple forms of ϕ can be as follows

$$\phi_1(S_i, R_j) = \begin{cases} 1 & \text{if } S_i = R_j \\ 0 & \text{otherwise.} \end{cases} \quad (3.11)$$

$$\phi_2(S_i, R_j) = \begin{cases} 1 & \text{if } S_i \cap R_j \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (3.12)$$

Equation 3.11 implies that two entities are the same if all the mentions are the same. And Equation 3.12 implies that two entities are the same if they have at least one common mention. Neither Equations 3.11 nor 3.12 has the needed discriminativity power. But we can define other entity similarity metrics which have this discriminativity power as follows

$$\phi_3(S_i, R_j) = |S_i \cap R_j| \quad (3.13)$$

$$\phi_4(S_i, R_j) = \frac{2 \times |S_i \cap R_j|}{|S_i| + |R_j|} \quad (3.14)$$

Recall and precision of $CEAF^M$ metric are defined using ϕ_3 :

$$Recall = \frac{\sum_{S \in S^*} \phi_3(S, g(S))}{\sum_i \phi_3(S_i, S_i)} \quad (3.15)$$

$$Precision = \frac{\sum_{S \in S^*} \phi_3(S, g(S))}{\sum_i \phi_3(R_i, R_i)} \quad (3.16)$$

Where S^* is a set of S -subsets which are generated by intersecting S and the corresponding response sets. For $CEAF^M$, $\sum_i \phi_3(S_i, S_i)$ is equal to the number of markables in key, and $\sum_i \phi_3(R_i, R_i)$ is the number of markables in response. And as the number of markables in key and response is equal, the recall and precision formulas produce the same score.

If in Formulations 3.15 and 3.16 we replace ϕ_3 with ϕ_4 , then we will have recall and precision of $CEAF^E$ metric. For $CEAF^E$, $\sum_i \phi_3(S_i, S_i)$ is equal to the number of entities in key, and $\sum_i \phi_3(R_i, R_i)$ the number of entities in response. And as the number of entities in key and response can be different, the recall and precision formulas can produce different scores.

Regarding the singletons, $CEAF^M$ metric has the same problem as B^3 metric. In other words, when the corpus has a high amount of singleton entities, it does not have enough discriminativity power. The primary drawback of $CEAF^E$ is that it assigns equal weights to all entities, regardless of the number of markables they have [Recasens and Hovy, 2010]. This is similar to MUC metric's problem.

Example: using the same setting presented in Figures 3.3-3.6, recall, precision and F_1 of $CEAF^M$ and $CEAF^E$ are:

Response Partition 1:

$$CEAF_{F_1}^M = CEAF_{recall}^M = CEAF_{precision}^M = \left(\frac{4}{9} + \frac{0}{9} + \frac{3}{9}\right) = 77.8\%$$

$$CEAF_{recall}^E = \left(\frac{1}{3} + \frac{0}{3} + \frac{0.75}{3}\right) = 58.3\%$$

$$CEAF_{precision}^E = \left(\frac{1}{2} + \frac{0}{2} + \frac{0.75}{2}\right) = 87.5\%$$

$$CEAF_{F_1}^E = \frac{2 \times 58.3 \times 87.5}{58.3 + 87.5} = 70.0\%$$

Response Partition 2:

$$CEAF_{F_1}^M = CEAF_{recall}^M = CEAF_{precision}^M = \left(\frac{4}{9} + \frac{0}{9} + \frac{0}{9}\right) = 44.4\%$$

$$CEAF_{recall}^E = \left(\frac{61.5}{3} + \frac{0}{3} + \frac{0}{3}\right) = 20.5\%$$

$$CEAF_{precision}^E = \left(\frac{61.5}{1} + \frac{0}{1} + \frac{0}{1}\right) = 61.5\%$$

$$CEAF_{F_1}^E = \frac{2 \times 20.5 \times 61.5}{20.5 + 61.5} = 30.8\%$$

Response Partition 3:

$$CEAF_{F_1}^M = CEAF_{recall}^M = CEAF_{precision}^M = \left(\frac{1}{9} + \frac{1}{9} + \frac{1}{9}\right) = 33.3\%$$

$$CEAF_{recall}^E = \left(\frac{40}{3} + \frac{66.6}{3} + \frac{50}{3}\right) = 52.2\%$$

$$CEAF_{precision}^E = \left(\frac{40}{9} + \frac{66.6}{9} + \frac{50}{9}\right) = 17.4\%$$

$$CEAF_{F_1}^E = \frac{2 \times 52.2 \times 17.4}{52.2 + 17.4} = 26.1\%$$

3.7.4 BLANC

BLANC has been recently introduced by Recasens and Hovy [2010]. It is based on Rand index, whose value is between 0 and 1. If the key and response partitions do not agree in regard to any pair of markables, the Rand index will be 0. And if they are identical, it will be 1.

$$Rand_{index} = \frac{N_{00} + N_{11}}{N(N-1)/2}$$

where N_{00} is the number of the correctly ignored disreferent pairs, and N_{11} is the number of the correctly generated coreferent pairs. N is the number of all possible pairs.

In addition to N_{00} and N_{11} these are two numbers which are ignored in the Rand index: (i) N_{10} as the incorrectly ignored coreferent pairs, and (ii) N_{01} as the

incorrectly generated disreferent pairs. These values are used for computation of BLANC. It is clear that

$$N = N_{00} + N_{01} + N_{10} + N_{11}$$

BLANC consists of two main parts:

Attraction: an indication of how well a system attracts the coreferent pairs.

Repulsion: an indication of how well a system repulses the disreferent pairs.

The attraction recall, precision and F_1 measures are calculated as

$$\begin{aligned} \textit{Attraction}_{\textit{recall}} &= \frac{N_{11}}{N_{11} + N_{10}} \\ \textit{Attraction}_{\textit{precision}} &= \frac{N_{11}}{N_{11} + N_{01}} \\ \textit{Attraction}_{F_1} &= \frac{2 \times \textit{Attraction}_{\textit{recall}} \times \textit{Attraction}_{\textit{precision}}}{\textit{Attraction}_{\textit{recall}} + \textit{Attraction}_{\textit{precision}}} \end{aligned}$$

And the repulsion recall, precision and F_1 measures are calculated as

$$\begin{aligned} \textit{Repulsion}_{\textit{recall}} &= \frac{N_{00}}{N_{00} + N_{01}} \\ \textit{Repulsion}_{\textit{precision}} &= \frac{N_{00}}{N_{00} + N_{10}} \\ \textit{Repulsion}_{F_1} &= \frac{2 \times \textit{Repulsion}_{\textit{recall}} \times \textit{Repulsion}_{\textit{precision}}}{\textit{Repulsion}_{\textit{recall}} + \textit{Repulsion}_{\textit{precision}}} \end{aligned}$$

Now we can define recall, precision and F_1 of BLANC as follows:

$$\begin{aligned} \textit{BLANC}_{\textit{recall}} &= \frac{\textit{Attraction}_{\textit{recall}} + \textit{Repulsion}_{\textit{recall}}}{2} \\ \textit{BLANC}_{\textit{precision}} &= \frac{\textit{Attraction}_{\textit{precision}} + \textit{Repulsion}_{\textit{precision}}}{2} \\ \textit{BLANC}_{F_1} &= \frac{\textit{Attraction}_{F_1} + \textit{Repulsion}_{F_1}}{2} \end{aligned}$$

Example: we calculate the scores of the examples presented in Figures 3.3-3.6 for BLANC as follows:

Response Partition 1:

$$BLANC_{recall} = ((\frac{10}{10+0}) + (\frac{20}{20+6}))/2 = 88.5\%$$

$$BLANC_{precision} = ((\frac{10}{10+6}) + (\frac{20}{20+0}))/2 = 81.2\%$$

$$BLANC_{F_1} = ((\frac{2 \times 100 \times 62.5}{100 + 62.5}) + (\frac{2 \times 76.9 \times 100}{76.9 + 100}))/2 = 84.7\%$$

Response Partition 2:

$$BLANC_{recall} = ((\frac{10}{10+0}) + (\frac{0}{0+26}))/2 = 50.0\%$$

$$BLANC_{precision} = ((\frac{10}{10+26}) + (\frac{0}{0+0}))/2 = 13.9\%$$

$$BLANC_{F_1} = ((\frac{2 \times 100 \times 27.8}{100 + 27.8}) + (\frac{2 \times 0 \times 0}{0 + 0}))/2 = 21.7\%$$

Response Partition 3:

$$BLANC_{recall} = ((\frac{0}{0+10}) + (\frac{26}{26+0}))/2 = 50.0\%$$

$$BLANC_{precision} = ((\frac{0}{0+0}) + (\frac{26}{26+10}))/2 = 36.1\%$$

$$BLANC_{F_1} = ((\frac{2 \times 0.0 \times 0}{0 + 0}) + (\frac{2 \times 100 \times 72.2}{100 + 72.2}))/2 = 41.9\%$$

3.7.5 Evaluation Problem

Some systems report only one of the above mentioned metrics. In my view, such an evaluation is insufficient. For example, taking the OntoNotes test set (see Section 4.7.3), if all markables in each document are placed into one cluster, the MUC F_1 score is equal to 45.2%, which is higher than most published results; the $B^3 F_1$ score of 6.7% in this case shows that, in reality, putting all markables in a single cluster is a poor strategy [Recasens et al., 2010].

Recasens et al. [2010] showed that there is a significant positive correlation between B^3 and $CEAF^M$ (Pearson's $r = 0.91$, $p < 0.01$). Therefore, reporting only the two correlated metrics B^3 and $CEAF^M$ is insufficient as well. For example, taking again the OntoNotes test set, if each markable forms a separate

chain, the B^3 and $CEAF^M$ F_1 scores are equal to 83.2% and 71.2%, which incorrectly suggests that performance is good; but MUC F_1 score in this case is equal to 0, demonstrating that this “singletons-only” strategy is bad. To avoid this, a solution is to use one of the following combinations: (i) MUC and B^3 , or (ii) MUC and $CEAF^M$. Roughly speaking, systems that incorrectly generate large clusters are penalized by the B^3 and $CEAF^M$ metrics, and systems that incorrectly generate singletons are penalized by the MUC metric.

3.8 Conclusion

This chapter reviewed the literature and related work concerning coreference resolution.

First, the problem of coreference resolution was described. Four categories of coreference resolution were reviewed: (i) Rule-based approaches, with Hobbs [Hobbs, 1986], CogNIAC [Baldwin, 1996], Mitkov [Mitkov, 1998] and Multi-Pass-Sieve [Raghunathan et al., 2010] methods presented; (ii) Supervised approaches, with the decision tree [Soon et al., 2001], generative probabilistic [Ge et al., 1998] and ranking approaches [Denis and Baldridge, 2007a] presented; (iii) Unsupervised approaches, where a nonparametric Bayesian model [Haghighi and Klein, 2007] and a Markov logic model [Poon and Domingos, 2008] have been presented; and (iv) Semi-supervised approaches, where Self-Training [Ng and Cardie, 2003] and Co-Training [Müller et al., 2002] methods were reviewed.

I also presented the most commonly used metrics for the evaluation of coreference resolution, each accompanied with a running example. I also showed that it is insufficient to report on only one of the above mentioned metrics.

Chapter 4

Supervised Coreference Resolution

4.1 Introduction

In this chapter, SUCRE, a supervised system for coreference resolution will be introduced. It is modular, flexible and language independent. Its modular architecture makes it possible to use any externally available classification method in its learning/classification core.

The modular architecture of SUCRE contains three main modules: preprocessing, pair estimation and chain estimation. The details of these modules will be presented by breaking them down into separate interchangeable modules.

In order to evaluate this system I participated in two international shared tasks on coreference resolution, namely, (i) SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010]; and (ii) CoNLL-2011 Shared Task on Modeling Unrestricted Coreference in OntoNote [Pradhan et al., 2011b]. The results of SUCRE at these shared tasks shows that SUCRE has been optimized in a way that achieves good results on the different scoring metrics and multiple languages. This good performance is a clear demonstration of the strength of the system and general validity of my design.

SUCRE has been developed to provide a more flexible method for feature engineering of coreference resolution. It has a novel approach to model an unstructured text corpus in a structured framework uses a relational data model

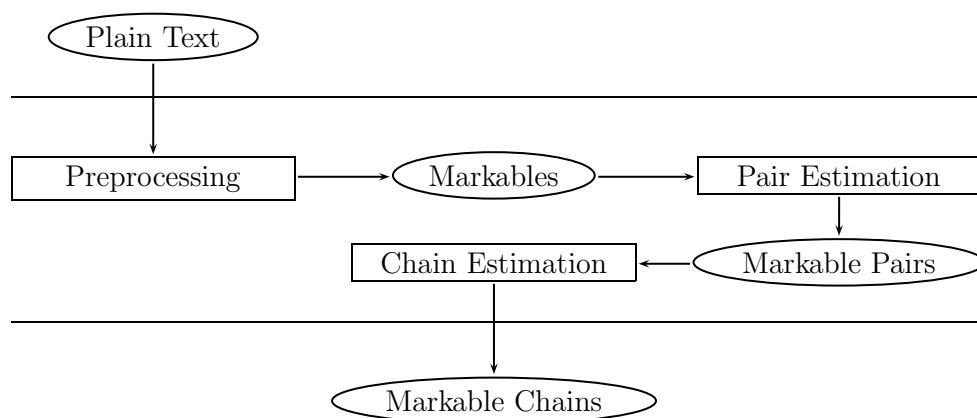


Figure 4.1: Top Level Architecture of SUCRE.

and a regular feature definition language to define and extract the features. This will be presented in detail in Chapter 5.

This chapter is organized as follows. System architecture of SUCRE is presented in section 4.2. In 4.3, 4.4 and 4.5, the preprocessing, pair estimation and chain estimation modules will be presented. Section 4.6 presents a brief evaluation of SUCRE, comparing it with a comparable system. In 4.7 and 4.8, the shared tasks in which SUCRE has successfully participated will be reported. And finally section 4.9 concludes this chapter.

4.2 System Architecture

Figure 4.1 illustrates the top level architecture of our supervised coreference resolution system SUCRE. The oval nodes are data, and the box nodes are processes.

This top level architecture provides a clear separation between three main functional components of the systems, namely **(i)** Preprocessing; **(ii)** Pair Estimation; and **(iii)** Chain Estimation. Also, it defines four main data forms that pass through the functional components: **(i)** Plain Text; **(ii)** Markables; **(iii)** Markable Pairs; and **(iv)** Markable Chains. This shows that in its top level architecture, SUCRE is a *modular* system. According to the definition of modularity, a modular system

- is composed of **separate, interchangeable** components (this is provided

by breaking down the system into modules).

- consists of modules that each accomplish one function and contain everything necessary to accomplish this.
- improves maintainability by enforcing **logical boundaries** between components.

As will be shown below, each of the above mentioned functional components of SUCRE is broken down into separate interchangeable modules, implying that the framework in its lower levels is also *modular*.

Furthermore, in order to provide the needed logical boundaries between components, SUCRE uses a relational data model whose details will be presented in Chapter 5. At the core of this relational data model, there are three relations (i.e. tables): Token, Markable and Markable Pair, which are as follows:

- **Token:**
(Token-ID, Sentence-ID, Token-String, Attribute-Vector)
- **Markable:**
(Markable-ID, First-Token-ID, Last-Token-ID, Head-Token-ID, Attribute-Vector)
- **Markable Pair:**
(First-Markable-ID, Second-Markable-ID, Feature-Vector)

This approach to data provides clear, well-defined logical boundaries. Also, it makes it possible that each module contains everything necessary to accomplish its function. This means that SUCRE is a highly modular system.

Using the above described data model, we were able to integrate a visualization method of coreference data based on Self Organizing Maps (SOMs) [Burkovski et al., 2010] into our framework. This method can be seen as a logically separated subsystem with these objectives: (i) to enable the user to explore the high dimensional feature space; (ii) to enable the user to annotate many links at once; and (iii) to allow the user to judge the quality of the features and to explore nodes with mixed coreference information. A brief introduction to the visualization approach is presented in appendix A.

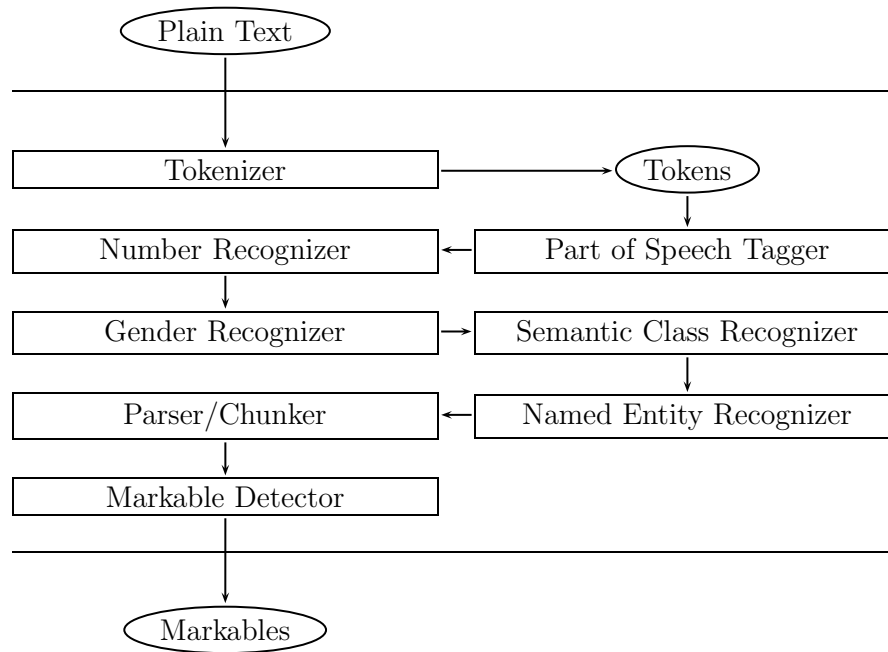


Figure 4.2: Preprocessing Pipeline

4.3 Preprocessing

The preprocessing pipeline accepts plain text as the input and produces markables. It is clear from the above described data model that markables are defined according to the token and markable tables.

Figure 4.2 presents a possible architecture of the preprocessing pipeline. I use the word “possible”, because it is not necessary to have all modules in the pipeline. The two most important components are **Tokenizer** (to generate token table), and **Markable Detector** (to generate markable table). The other modules are to extract/generate the attributes of tokens (e.g. **Number** and **Gender Recognizer**) or help to detect markables (e.g. **Parser**).

4.3.1 Tokenizer

Tokenization is the process of converting a sequence of characters (i.e. plain text) into a sequence of tokens. It has nothing to do with combinations of tokens. A

token is a string of characters which can be of different types (e.g. word and punctuation). A token is the basic block for processing an input text stream.

Tokenization is usually a trivial task that any natural language processing system typically performs as the first step.

4.3.2 Part of Speech Tagger

Part of speech is one of the most important attributes which can be used for feature definition of coreference resolution. Part of speech tagging is performed after tokenization. The modular architecture of our system allows the use of any part of speech tagger. All that is needed is that the output of tagger is written in the token table, where a part of speech tag is an attribute of a token. In the current implementation of SUCRE, “TreeTagger” is used as the part of speech tagger [Schmid, 1994].

4.3.3 Number Recognizer

Number recognition can be partially performed using part of speech tags. Usually a part of speech tagger indirectly detects noun and verb number (if there is a meaningful one). And for pronouns it can be done simply using fixed rules (e.g. “he” → “singular” , “we” → “plural”).

4.3.4 Gender Recognizer

Gender recognition in SUCRE uses the gender data that was created by Bergsma and Lin [2006]. The gender data consist of four numbers for a noun phrase relating to the number of times that the noun phrase has been seen as male, female, neuter or plural. These numbers are taken and a simple comparison is performed as to which case has the maximum value to determine the corresponding gender.

4.3.5 Semantic Class Recognizer

SUCRE includes a semantic class recognizer based on WordNet [Fellbaum, 1998].

The semantic class recognizer looks at the synonyms of the token and if one of them is in the predefined keyword set, it takes it as the corresponding semantic class of the token. These are the keywords which are pre-defined in SUCRE: act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, natural object, person, natural phenomenon, plant, possession, process, quantity, relation, shape, state, substance and time.

4.3.6 Named Entity Recognizer

A good named entity recognizer plays an important role in coreference resolution. Since in SUCRE named entity is an attribute of markable, this step of preprocessing can be done together with the markable detection step. Recognized named entities can be used for improving the quality of markable detection so that if a noun phrase is a named entity then all other noun phrases which include this named entity or are included in this named entity can be deleted from the list of markable candidates.

4.3.7 Parser

Typically a parser produces tree structures which can be used for two purposes: (i) syntactic feature extraction, and (ii) markable detection. SUCRE accepts the Penn Treebank notation of parse trees.

There are different types of syntactic features which can be defined for markables. There is a class of features in which the path between two markables from the parse tree is extracted. One can then look to see if this path matches the already learned patterns.

Figure 4.3 presents an example of such features. Each up arrow means that the path is going up in the parse tree, whereas each down arrow represents a downward path.

For example, feature $NP \uparrow S \downarrow VP \downarrow SBAR \downarrow S \downarrow NP \downarrow PRP$ means that the first markable is a noun phrase (i.e. NP tag) that has a direct path to the sentence root (i.e. S tag), and from root the path goes down through $VP, SBAR, S, NP$ to the second markable, which is a pronoun (i.e. PRP tag).

Figure 4.3 shows the most frequent coreference paths seen in the OntoNotes corpus [Pradhan et al., 2007b].

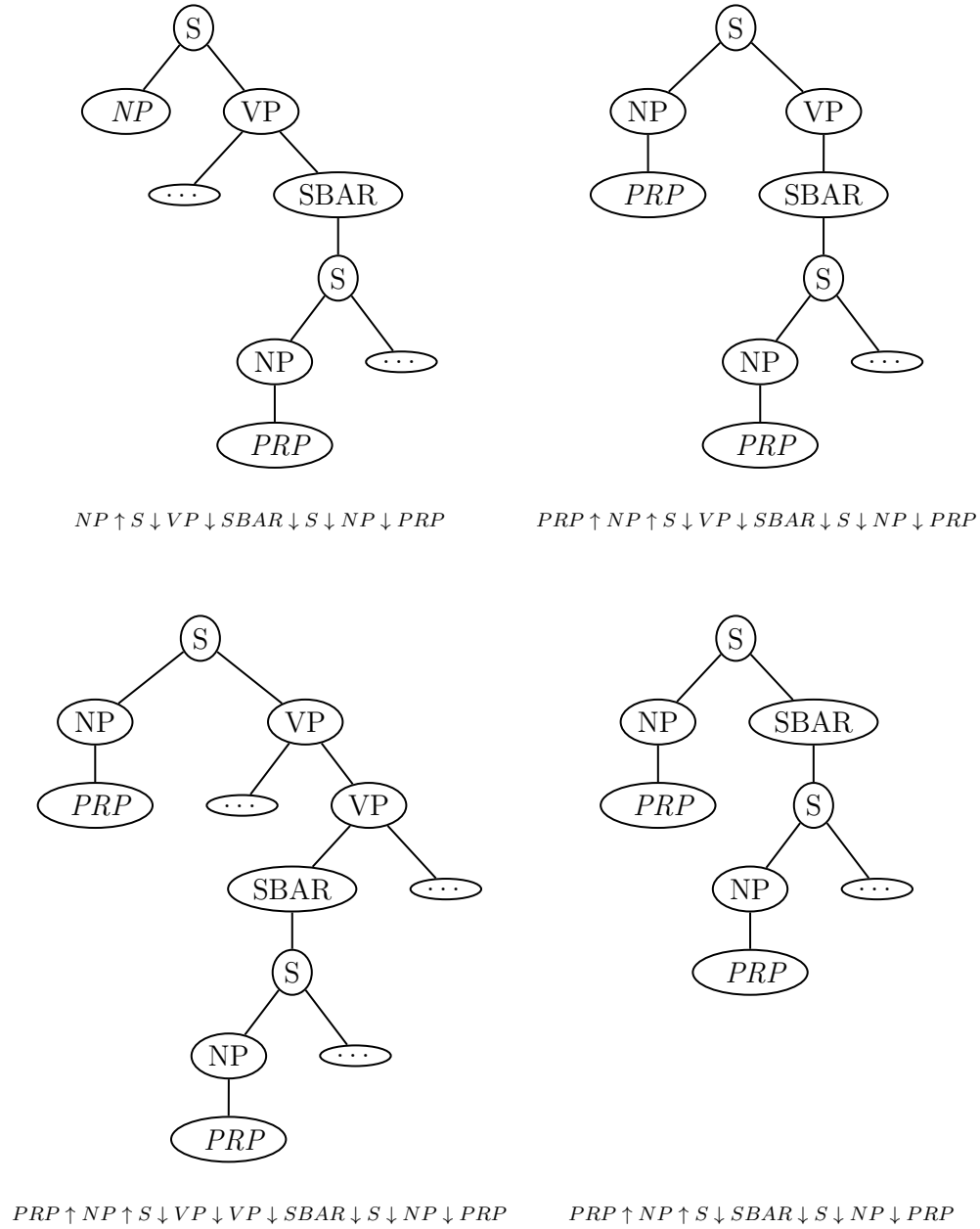


Figure 4.3: Most frequent parse tree paths in OntoNotes.

4.3.8 Markable Detector

Markable detection is considered a sub-task of coreference resolution. From an annotation point of view, markables are of two types:

True markables: those extracted from the answer keys.

System markables: those automatically detected by the system.

In addition, there is another property that can divide markables into the two categories **anaphoric markables** vs. **non-anaphoric markables**. An anaphoric markable is a markable that is coreferent with at least one other markable, whereas a non-anaphoric markable has no coreferent.

There are different methods of markable detection. For example, the easiest way a markable can be detected is by looking at only part of speech tags. But such a method has a poor performance. Another example is to take all noun phrases which are detected by a text chunker (see 2.7.1).

A common approach to the sub-task of markable detection is to extract noun phrases from parse trees. For example, consider this sentence “we did the very best we could in these situations”, whose parse tree (using phrase structure grammar) is presented in Figure 4.4. There are 4 noun phrases (shown in italic) which can be extracted as markable: (i) “we”; (ii) “the very best”; (iii) “we”; and (iv) “in these situations”.

Figure 4.4 presents a simple case in which there are no nested noun phrases. A problem, however, arises when there are nested noun phrases in the parse tree: in the case of nested noun phrases, which of the following approaches should be taken for markable detection?

1. all possible noun phrases
2. only lower bound noun phrases (i.e. a subset of first case).
3. only upper bound noun phrases (i.e. a subset of first case).
4. a combination of above cases (i.e. a subset of first case).

Figure 4.5 presents an example of nested noun phrases. The output markables of the above mentioned approaches for this example are listed as follows:

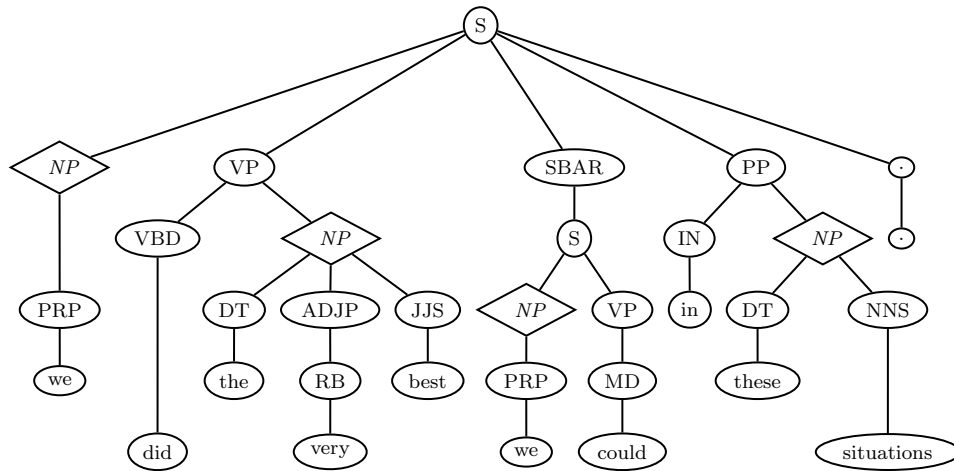


Figure 4.4: Parse Tree for: “we did the very best we could in these situations.”

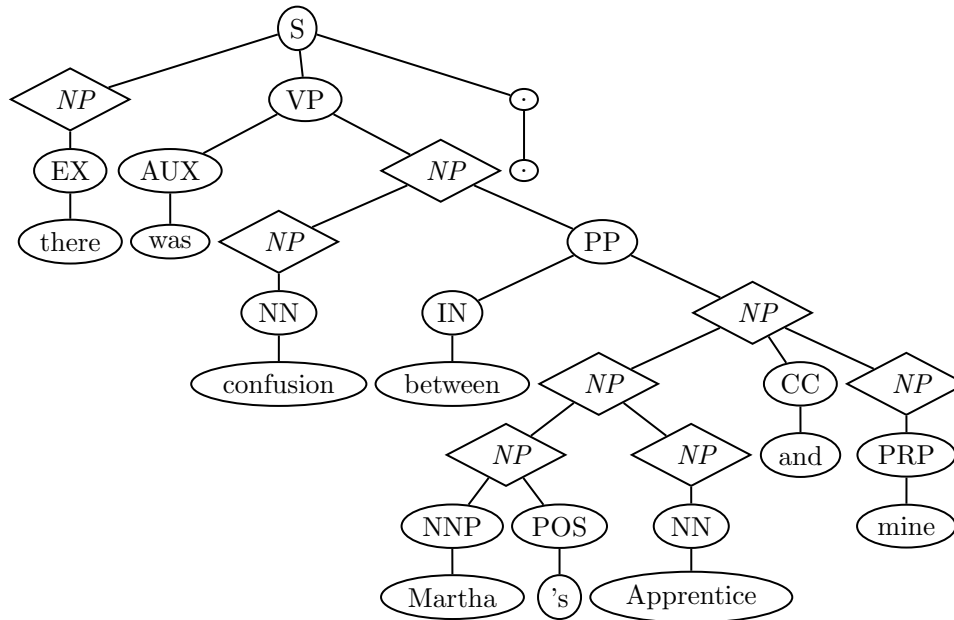


Figure 4.5: Parse Tree for: “there was confusion between Martha’s Apprentice and mine.”

1. “there”, “confusion”, “Martha’s”, “Apprentice”, “mine”, “Martha’s Apprentice”, “Martha’s Apprentice and mine” and “confusion between Martha’s Apprentice and mine”.
2. “there”, “confusion”, “Martha’s”, “Apprentice” and “mine”.
3. “there” and “confusion between Martha’s Apprentice and mine”.
4. “Martha’s”, “Martha’s Apprentice” and mine”.

Figure 4.6 shows the markable detection algorithm that extracts all possible noun phrases from parse trees of phrase structure grammar (PSG).

Figure 4.7 presents an example of a dependency parse tree. Figure 4.9 presents the markable detection algorithm that extracts markables from parse trees of dependency grammar (DG). Figure 4.8 shows the output of the algorithm presented in Figure 4.9 on the example illustrated in Figure 4.7.

4.4 Pair Estimation

Pair estimation is done in two different forms (i) learning (training); and (ii) classification (test).

4.4.1 Learning

Figure 4.10 shows the internal architecture of the pair estimation module for learning. There are four functional sub-modules which will be presented in detail.

Pair Generation

For training, the system generates a positive training instance for each adjacent coreferent markable pair, and negative training instances for a markable m and all markables disreferent with m that occur before m [Soon et al., 2001].

This method was able to be applied on all markables either starting from the end of the document (backward pair generation) or from its beginning (forward pair generation). Figure 4.11 illustrates each of these two variants of pair generation method as pseudo code.

Markable_Detection_PSG_A (W_1, W_2, \dots, W_n)

1. A markable M is presented by a set of three words:
Begin (M_b), End (M_e) and Head (M_h).
2. Let DM be the set of detected markables.
3. Let T_i be the node i in the parse tree with label L_i
(if node is a word then L_i is equal to W_i).
4. Start from parse tree root T_r : $Find_Markables(T_r, L_r, DM)$

Find_Markables(T, L, DM)

1. If L is a noun phrase, then extract the markable M (M_b, M_e, M_h):
 - (a) $M_b = Noun_Phrase_Begin(T, L)$
 - (b) $M_e = Noun_Phrase_End(T, L)$
 - (c) $M_h = Noun_Phrase_Head(T, L)$
2. Add the markable M to the set of detected markables DM .
3. Repeat for all T_i the daughters of T : $Find_Markables(T_i, L_i, DM)$

Noun_Phrase_Begin(T, L)

If T has no daughter then return L ;
else set T_b to the first daughter of T and return $Noun_Phrase_Begin(T_b, L_b)$.

Noun_Phrase_End(T, L)

If T has no daughter then return L ;
else set T_b to the last daughter of T and return $Noun_Phrase_End(T_b, L_b)$.

Noun_Phrase_Head(T, L)

If T has no daughter then return L ;
else set T_h to the biggest noun phrase daughter of T and return $Noun_Phrase_Head(T_h, L_h)$.

Figure 4.6: Markable Detection from Phrase Structures.

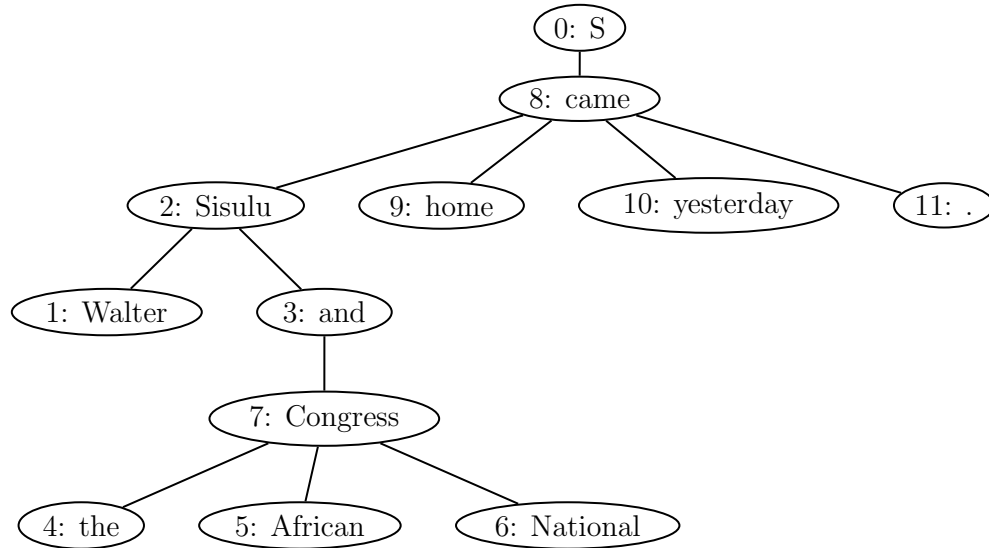


Figure 4.7: Dependency Structure of “Walter Sisulu and the African National Congress came home yesterday.”

No	Token	POS	Dependency	Role
1	Walter	NNP	2	NAME
2	Sisulu	NNP	8	SBJ
3	and	CC	2	COORD
4	the	DT	7	NMOD
5	African	NNP	7	NAME
6	National	NNP	7	NAME
7	Congress	NNP	3	CONJ
8	came	VBD	0	sentence
9	home	NN	8	DIR
10	yesterday	NN	8	TMP
11	.	.	8	P

Detected markable 1: “Walter *Sisulu*”

Detected markable 2: “Walter *Sisulu* and the African National Congress”

Detected markable 3: “the African National *Congress*”

Detected markable 4: “*home*”

Detected markable 5: “*yesterday*”

Figure 4.8: Detected markables for above example.

Markable_Detection_DG (W_1, W_2, \dots, W_n)

1. A markable M is presented by a set of three words:
Begin (M_b), End (M_e) and Head (M_h).
2. Let DM be the set of detected markables.
3. Let POS_j be the part of speech of the word W_j .
4. Proceed through the words. For each word W_j :
if POS_j is equal to NNP and POS_{j+1} is not equal NNP or
if POS_j is equal to NN or PRP ,
then extract the markable M :
 - (a) Set the begin word of the markable:
 $M_b = \text{Dependency_Head_Begin}(1, j)$
 - (b) Set the end word of the markable:
 $M_e = \text{Dependency_Head_End}(j, n)$
 - (c) Set the head word of the markable:
 $M_h = W_j$
 - (d) Add the markable M to the set of detected markables DM .

Dependency_Head_Begin (x, y)

If $x == y$ return then W_x ;
 else if W_y is the head of W_x then return W_x ;
 else return $\text{Dependency_Head_Begin}(x + 1, y)$.

Dependency_Head_End (x, y)

If $x == y$ then return W_y ;
 else if W_x is the head of W_y then return W_y ;
 else return $\text{Dependency_Head_End}(x, y - 1)$.

Figure 4.9: Markable Detection from Dependency Parse Information.

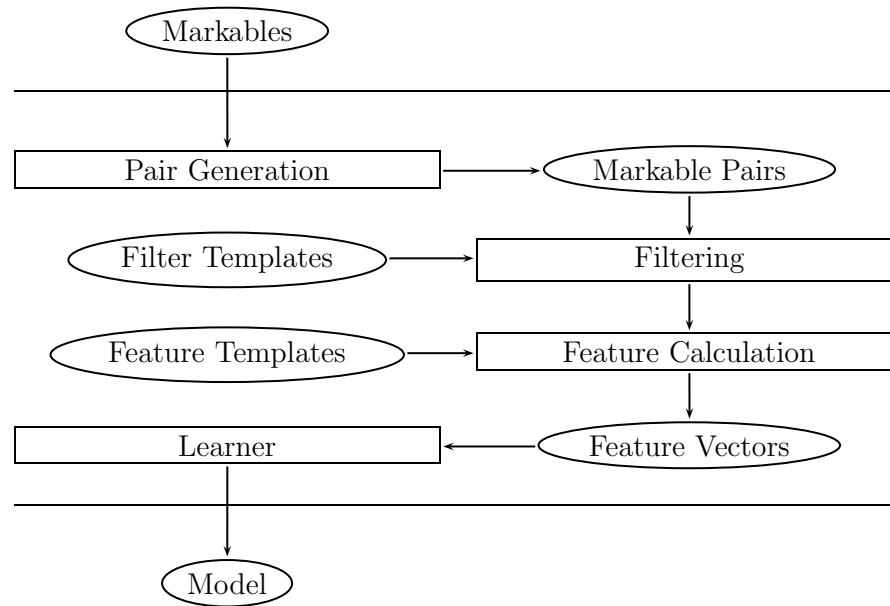


Figure 4.10: Pair Estimation in SUCRE: Learning

Filtering

Filters are high-accuracy pair features which can be used directly to filter the markable pairs while generating the pairs or in the clustering step. For example, number or natural gender disagreements are very good indicators of disreference and can be used as filters. Table 4.1 presents some examples of such features.

Feature Calculation

In SUCRE, pair features are used for learning. For pair feature extraction, the head words of the markables are usually used, but in some cases the head word may not be a suitable choice. For example, consider the two markables *the books* and *a book*. In both cases *book* is the head word, but to distinguish which markable is definite and which indefinite, the article must be taken into account. Now consider the two markables *the students from Germany* and *the students from France*. In this case, the head words and the first four words of each markable are the same but they cannot be coreferent; this can be detected only by looking at the last words. Sometimes we need to consider all words in the two markables,

Pair_Generation_Backward (M_1, M_2, \dots, M_n)

1. initialize $k = n$
2. set $i = k - 1$ and $j = k$
3. if pair (M_i, M_j) is coreferent:
 - (a) add it to the positive instances
 - (b) for all M_l where $i < l < j$:
add the pairs (M_l, M_j) to the negative instances
 - (c) set $k = i$, if $k > 1$ goto step 2
4. else (i.e. pair (M_i, M_j) is disreferent):
 - (a) add it to the negative instances
 - (b) set $i = i - 1$, if $i \geq 1$ goto step 3
 - (c) set $k = k - 1$, if $k > 1$ goto step 2

Pair_Generation_Forward (M_1, M_2, \dots, M_n)

1. initialize $k = 1$
2. set $i = k$ and $j = k + 1$
3. if pair (M_i, M_j) is coreferent:
 - (a) add it to the positive instances
 - (b) for all M_l where $i < l < j$:
add the pairs (M_l, M_j) to the negative instances
 - (c) set $k = k + 1$, if $k < n$ goto step 2
4. else (i.e. pair (M_i, M_j) is disreferent):
 - (a) set $j = j + 1$, if $j \leq n$ goto step 3
 - (b) set $k = k + 1$, if $k < n$ goto step 2

Figure 4.11: Backward and Forward Pair Generation Algorithms.

Feature	Example
Semantic Class	<u>Berlin</u> - <u>1989</u>
Natural Gender	<u>Mary</u> - <u>he</u>
Grammatical Gender of Pronouns	<u>she</u> - <u>he</u>
Number	the <u>men</u> - the <u>man</u>
Person	<u>I</u> - <u>you</u>

Table 4.1: Filter Examples.

or even define an atomic feature for a markable.

To cover all such cases the relational model described before was used. More details will be presented in Chapter 5.

Two examples of pair features are as follows:

- there is at least one exact match between the words of the markables, and the head words of both are nouns.
- two markables are in the same sentence and the type of the second markable head word is reflexive.

Learner

There are four classifiers integrated in SUCRE: Decision-Tree, Naive-Bayes, Support Vector Machine [Joachims, 2002] and Maximum-Entropy [Tsuruoka, 2006].

When I compared these classifiers, the best results (which will be reported in Sections 4.7 and 4.8) were achieved with Decision-Tree.

Roughly speaking, there is a trade-off between feature engineering (feature extraction) and decision making [Alpaydin, 2010]. Better feature extraction leads to an easier classification task.

Also, with a good enough classification method, we do not need any additional feature extraction method. In other words, a sufficient classifier selects its features internally. There is, however, a disadvantage in that, in the case of any change in the feature space properties, the classifier will have to be redesigned.

To prevent this disadvantage and also to bring about more modularity, feature engineering was performed externally, i.e. in the decision tree which was

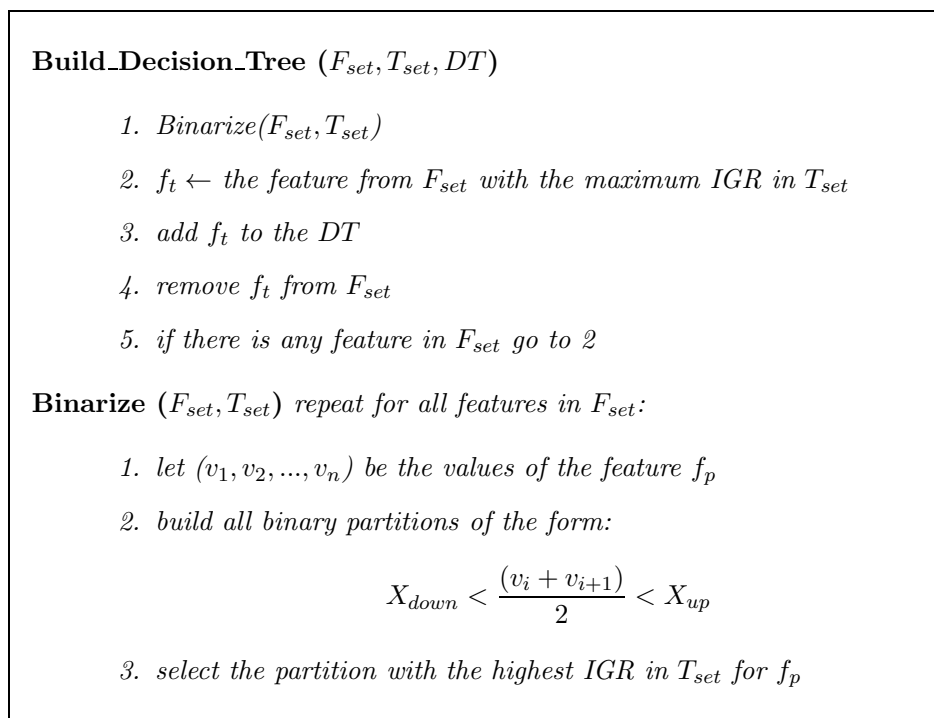


Figure 4.12: Decision Tree.

implemented no internal feature selection was performed. As will be seen later in chapter 5, feature extraction can be done externally using my novel feature engineering method based on relational data model and a feature definition language.

I used Information Gain Ratio (see 2.21) for building the decision tree. Figure 4.12 presents the simplified form of the algorithm used for building the decision tree using Information Gain Ratio. F_{set} is the feature set, T_{set} the training set and DT the decision tree. In Step 2 a decision rule for feature f_t was added wherever a node has no child and is not distinguished as a leaf. A leaf is a node without child, where the number of positive instances is more than 50%, or the number of negative instances is more than 99%.

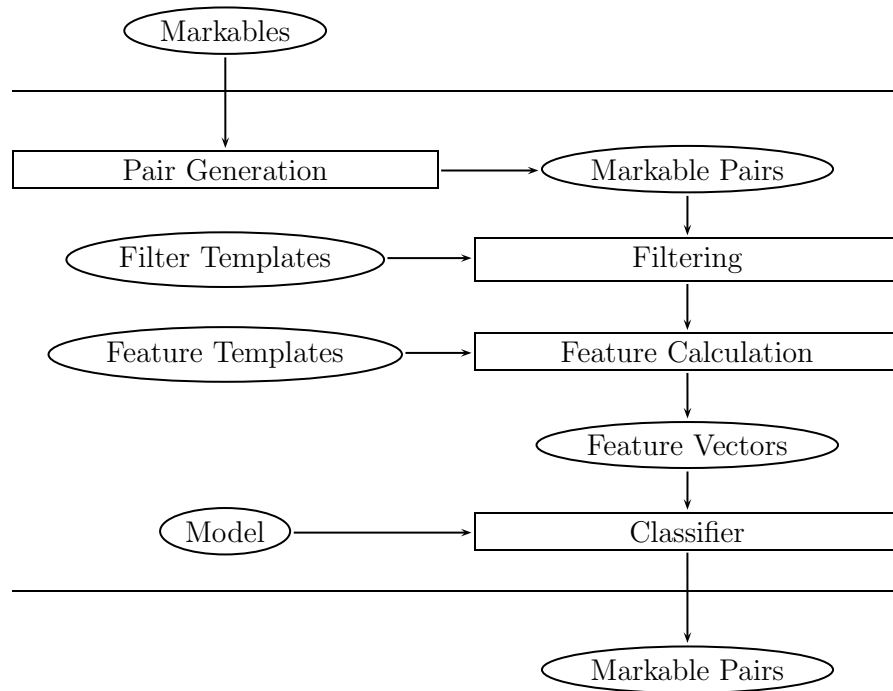


Figure 4.13: Pair Estimation in SUCRE: Classification

4.4.2 Classification

Pair Generation

Logically, for classification all possible markable pairs inside a document should be generated. As it is not efficient, a limit should be applied. McEnery et al. [1997] showed that in 98.68% of cases the antecedent is within a 10-sentence window; hence, in the pair generation step, all possible pairs inside 10 sentences are generated.

Filtering and Feature Calculation

These two steps are identical to the corresponding steps in the learning stage.

Classifier

As was mentioned before, a decision tree was used for classification. In each leaf of the tree there are positive and negative instances and these make it possible to calculate the coreference probability as follows:

$$p(\text{coreference}) = \frac{N_{\text{positive}}}{N_{\text{positive}} + N_{\text{negative}}}$$

4.5 Chain Estimation

In decoding, the coreference chains are created. SUCRE introduces a novel method, similar to the best-first clustering method, for this purpose. It searches for the best predicted antecedent above a threshold from right-to-left starting from the end of the document. Figure 4.14 presents the details of our clustering algorithm.

We call this method thresholded best-first clustering. Compared to the clustering method proposed by Wagstaff [2002], the thresholded best-first clustering method uses a decreasing threshold for completing the clustering task. In this way it is possible to consider all candidates that are in the range above the threshold. For example, if antecedent M_1 has the probability 0.982 and M_2 has the probability 0.985, then by a threshold t equal to 0.98, both of them come into question, and the nearest one is selected.

Chain_Estimation (M_1, M_2, \dots, M_n)

1. Let t be the clustering threshold initialized with 1.
2. Let each markable M_i be in its own cluster $C_i : C_i = \{M_i\}$.
3. Proceed through the markables from the end of the document. For each markable M_j , consider each preceding markable M_i within a window of 10 sentences:
If $\text{Pair_Estimation}(M_i, M_j) \geq t$ then $C_i = C_i \cup C_j$.
4. Set $t = t - 0.01$
5. If $t \geq 0.5$ go to step 3.

Pair_Estimation (M_i, M_j):

If $\text{Filtering}(M_i, M_j) == \text{TRUE}$ then return 0; else return the probability of markable pair (M_i, M_j) being coreferent.

Filtering (M_i, M_j):

If one of the filtering rules for the markable pair (M_i, M_j) is true then return **TRUE**; else return **FALSE**.

Figure 4.14: Chain Estimation Algorithm.

	ACE-2		ACE2003		MUC-6	
	MUC- F_1	B ³ - F_1	MUC- F_1	B ³ - F_1	MUC- F_1	B ³ - F_1
Reconcile-2010	65.99	78.29	67.87	79.39	68.50	70.88
SUCRE	72.43	75.63	73.01	74.30	70.58	69.47

Table 4.2: Compare SUCRE and Reconcile-2010 on the ACE-2, ACE2003 and MUC-6 data sets.

4.6 SUCRE vs. Reconcile-2010

Table 4.2 shows B^3 and $MUC F_1$ -Scores of SUCRE and the recently published results of Reconcile-2010 [Stoyanov et al., 2010] on the following data sets:

ACE (Phase 2): I used two variants of ACE (Phase 2): ACE-2 and ACE2003 [Mitchell et al., 2003].

MUC-6: the data set from the sixth Message Understanding Conference (MUC-6) evaluation [Chinchor and Sundheim, 2003].

Our system has better MUC score (+6.44) but the B^3 score is better in Reconcile-2010 (-2.66). The results of SUP show that its performance is comparable to that of a state-of-the-art supervised system for coreference resolution.

4.7 SemEval-2010

SUCRE successfully participated in SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010]. In this section the task will briefly be described and the feature sets and results of SUCRE on this shared task reported.

4.7.1 Task

The task considered the intra-document coreference resolution of six different languages, namely: Catalan, Dutch, English, German, Italian and Spanish. The task was to identify which markables (noun phrases) in a text refer to the same discourse entity.

The task organizers named the main goal of the shared task as: “to perform and evaluate coreference resolution for six different languages with the help of other layers of linguistic information and using different evaluation metrics (MUC, B-CUBED, CEAF and BLANC)”.

In addition to the above-mentioned goal, we had others, including:

- The multilingual context made it possible to investigate the portability of coreference resolution systems across languages. The following questions were relevant:
 - To what extent can a general system which is portable to all six languages be implemented ?
 - How much language-specific tuning is needed?
 - Are there significant differences between Germanic and Romance languages?
 - Are there significant differences between the languages of the same family (e.g. Catalan and Spanish)?
- The additional layers of annotation made it possible to investigate how helpful morphology, syntax and semantics are for coreference resolution. The following questions were relevant:
 - How much preprocessing is needed?
 - What is the effect of the quality of the preprocessing on the performance of state-of-the-art coreference resolution systems?
 - Can the helpfulness of morphology and syntax information be compared?
 - Can the helpfulness of morphology and semantics information be compared?
 - Can the helpfulness of syntax and semantic information be compared?
- Four different evaluation metrics were used to compare the advantages and drawbacks of each. The following questions were relevant:
 - Are the rankings that they provide the same?

- Is there any correlation between them?
- Is it possible to optimize the systems under all four metrics at the same time?

4.7.2 System Description

Using SUCRE, we did the following steps for training:

1. convert data to the relational model.
2. detect markables.
3. generate markable pairs for training.
4. extract features, in a loop:
 - (a) define new features (either from scratch or by combination of existing features).
 - (b) evaluate the features using Information Gain Ratio.
 - (c) keep the promising features in the feature set.
 - (d) remove the non-performing features from the feature set.
 - (e) perform data analysis to determine which phenomena are not captured by the current feature set.
5. Train the decision tree.

And the following steps for the testing:

1. convert data to the relational model.
2. detect markables.
3. generate all markable pairs inside a window of 10 sentences.
4. calculate the feature values.
5. classify the markable pairs.
6. generate markable chains (cluster).

4.7.3 Data Sets

The description of the data sets that were used in the experiments is as follows:

Catalan:

The AnCora-Ca corpus [Recasens et al., 2010] consists of a Catalan treebank of 400K words, mainly from newspapers and news agencies. It has been manually annotated for arguments and thematic roles, predicate semantic classes, named entities, WordNet nominal senses, and coreference relations. We used 300K words for training, 50K words for development and 50K words for test.

German:

The data set comes from the TüBa-D/Z Treebank [Hinrichs et al., 2005]. It is a German newspaper corpus based on data taken from the daily issues of "die tageszeitung" (taz). The corpus has been hand-annotated with inflectional morphology, constituent structure, grammatical functions, and anaphoric and coreference relations. We used 415K words for training, 50K words for development and 50K words for test.

English:

Part of the OntoNotes Corpus Release 2.0 was used for English. The OntoNotes project is a hand-annotated corpus and is a collaborative effort between BBN Technologies, the University of Colorado, the University of Pennsylvania, and the University of Southern California's Information Sciences Institute. It contains structural information (syntax and predicate argument structure) and shallow semantics (word sense linked to an ontology, NEs and coreference). Here we used 100K words for training, 25K words for development and 24K words for test.

Spanish:

The AnCora-Es corpus [Recasens et al., 2010] consists of a Spanish treebank of 430K words, mainly from newspapers and news agencies. It has been manually annotated for arguments and thematic roles, predicate semantic classes, named

entities, WordNet nominal senses, and coreference relations. We used 330K words for training, 50K words for development and 50K words for test.

Italian:

The data set comes from the LiveMemories corpus, which is an Italian corpus under construction in the LiveMemories project [Rodriguez et al., 2010]. The corpus contains texts from Wikipedia, blogs, and news articles. It has been annotated with coreference, agreement, and named entity information on top of automatically parsed data. The size of training, development and test sub sets were 100K, 50K and 50K words, respectively.

Dutch:

The KNACK-2002 corpus [Hoste and De Pauw, 2006] contains 267 documents from the Flemish weekly magazine Knack. The documents have been manually annotated with coreference information on top of semi-automatically annotated part of speech tags, phrase chunks, and named entities. Here we used 60K words for training, 50K words for development and 50K words for test.

4.7.4 Feature Sets

A comprehensive set of features was defined for each language. The feature sets of all six languages is provided in appendix B.

4.7.5 Results

Tables 4.3 and 4.4 show results from SemEval-2010 Task 1 Coreference Resolution in Multiple Languages [Recasens et al., 2010]. Table 4.3 contains SUCRE’s results. Table 4.4 shows in each case the best system other than SUCRE (selected based on B^3 score). Four different evaluation metrics were used in SemEval: MUC [Vilain et al., 1995], B^3 [Bagga and Baldwin, 1998], CEAF [Luo, 2005] and BLANC [Recasens and Hovy, 2010]. Note that the test set only became publicly available after the conclusion of SemEval-2010. Thus, during the development of SUCRE, and in particular for feature engineering, the test set

	language	ca	de	en	es	it	nl
Gold Track	MD-Rec.	100	100	100	100	98.4	100
	MD-Prec.	100	100	100	100	98.4	100
	MD-F ₁	100	100	100	100	98.4	100
	CEAF-Rec.	68.7	72.9	74.3	69.8	66.0	58.8
	CEAF-Prec.	68.7	72.9	74.3	69.8	66.0	58.8
	CEAF-F ₁	68.7	72.9	74.3	69.8	66.0	58.8
	MUC-Rec.	54.1	74.4	68.1	52.7	48.1	65.7
	MUC-Prec.	58.4	48.1	54.9	58.3	42.3	74.4
	MUC-F ₁	56.2	58.4	60.8	55.3	45.0	69.8
	B ³ -Rec.	76.6	90.4	86.7	75.8	76.7	65.0
	B ³ -Prec.	77.4	73.6	78.5	79.0	76.9	69.2
	B ³ -F ₁	77.0	81.1	82.4	77.4	76.8	67.0
	BLANC-Rec.	72.4	78.2	77.3	67.3	54.8	69.5
	BLANC-Prec.	60.2	61.8	67.0	62.5	63.5	62.9
	BLANC-F ₁	63.6	66.4	70.8	64.5	56.9	65.3
Regular Track	MD-Rec.	75.9	79.3	78.4	74.9	84.6	78.0
	MD-Prec.	64.5	77.5	83.0	66.3	98.1	29.0
	MD-F ₁	69.7	78.4	80.7	70.3	90.8	42.3
	CEAF-Rec.	51.3	60.6	61.0	56.3	57.1	29.4
	CEAF-Prec.	43.6	59.2	64.5	49.9	66.2	10.9
	CEAF-F ₁	47.2	59.9	62.7	52.9	61.3	15.9
	MUC-Rec.	44.1	49.3	57.7	35.8	50.1	62.0
	MUC-Prec.	32.3	35.0	48.1	36.8	50.7	19.5
	MUC-F ₁	37.3	40.9	52.5	36.3	50.4	29.7
	B ³ -Rec.	59.6	69.1	68.3	56.6	63.6	59.1
	B ³ -Prec.	44.7	60.1	65.9	54.6	79.2	6.5
	B ³ -F ₁	51.1	64.3	67.1	55.6	70.6	11.7
	BLANC-Rec.	53.9	52.7	58.9	52.1	55.2	46.9
	BLANC-Prec.	55.2	59.3	65.7	61.2	68.3	46.9
	BLANC-F ₁	54.2	53.6	61.2	51.4	57.7	46.9

Table 4.3: Results of SUCRE. Bold F-Measures indicate that SUCRE performed better than the other participants. MD: Markable Detection, ca: Catalan, de: German, en:English, es: Spanish, it: Italian, nl: Dutch

CHAPTER 4. SUPERVISED COREFERENCE RESOLUTION

	language	ca	de	en	es	it	nl
Gold Track	MD-Rec.	100	100	100	100	N/A	N/A
	MD-Prec.	100	100	100	100	N/A	N/A
	MD-F ₁	100	100	100	100	N/A	N/A
	CEAF-Rec.	70.5	77.7	75.6	66.6	N/A	N/A
	CEAF-Prec.	70.5	77.7	75.6	66.6	N/A	N/A
	CEAF-F ₁	70.5	77.7	75.6	66.6	N/A	N/A
	MUC-Rec.	29.3	16.4	21.9	14.8	N/A	N/A
	MUC-Prec.	77.3	60.6	72.4	73.8	N/A	N/A
	MUC-F ₁	42.5	25.9	33.7	24.7	N/A	N/A
	B ³ -Rec.	68.6	77.2	74.8	65.3	N/A	N/A
	B ³ -Prec.	95.8	96.7	97.0	97.5	N/A	N/A
	B ³ -F ₁	79.9	85.9	84.5	78.2	N/A	N/A
	BLANC-Rec.	56.0	54.4	57.0	53.4	N/A	N/A
	BLANC-Prec.	81.8	75.1	83.4	81.8	N/A	N/A
	BLANC-F ₁	59.7	57.4	61.3	55.6	N/A	N/A
Regular Track	MD-Rec.	83.3	60.9	79.6	82.2	42.8	41.5
	MD-Prec.	82.0	57.7	68.9	84.1	80.7	29.9
	MD-F ₁	82.7	59.2	73.9	83.1	55.9	34.7
	CEAF-Rec.	57.5	50.9	61.7	58.6	35.0	20.5
	CEAF-Prec.	56.6	48.2	53.4	60.0	66.1	14.6
	CEAF-F ₁	57.1	49.5	57.3	59.3	45.8	17.0
	MUC-Rec.	15.2	10.2	23.8	14.0	35.3	6.7
	MUC-Prec.	46.9	31.5	25.5	48.4	54.0	11.0
	MUC-F ₁	22.9	15.4	24.6	21.7	42.7	8.3
	B ³ -Rec.	55.8	47.2	62.1	56.6	34.6	13.3
	B ³ -Prec.	76.6	54.9	60.5	79.0	70.6	23.4
	B ³ -F ₁	64.6	50.7	61.3	66.0	46.4	17.0
	BLANC-Rec.	51.3	50.2	50.9	51.4	57.1	50.0
	BLANC-Prec.	76.2	63.0	68.0	74.7	68.1	52.4
	BLANC-F ₁	51.0	44.7	49.3	51.4	59.6	32.3

Table 4.4: Results of best systems without SUCRE. Bold F-Measures indicate that another participant performed better than SUCRE.

was not used. Hence, SUCRE’s performance is a realistic assessment of the performance a coreference resolution system developed using our feature engineering framework would be able to achieve on unseen data.

In comparison with the other systems, SUCRE achieved better results in closed regular annotation track with English and German (for all metrics). Its results for closed gold and regular annotation tracks of all six languages are the best in MUC and BLANC scoring metrics. And if it is not the best in the other metrics, the best result is only slightly better than result of SUCRE. For example, its results for closed gold annotation track of English and German are very close to the best system in CEAF and B³ (CEAF: English -1.3 German -4.8, B³: English -2.1 German -4.8;).

This result shows that SUCRE has been optimized in a way that achieves good results on the four different scoring metrics. This good performance is a demonstration of the strength of our framework: this method of feature definition, extraction and tuning is uniform and can be optimized and applied to all languages and tracks. Our system development for Catalan, Dutch, Italian and Spanish was done in only eight days. The fact that such competitive results were able to be obtained in such a short time shows the potential power of a flexible and fast framework for relational feature engineering.

Results of SUCRE show a correlation between the MUC and BLANC scores (the best MUC and BLANC scores of all tracks). In my opinion, this correlation is not due to the high similarity between MUC and BLANC, but rather because of the balanced scores.

4.7.6 Evaluation Problem

As described in Section 3.7, we need different metrics because evaluation by considering only one metric could lead to misjudgment. Table 4.5 confirmed this by presenting the scores for two baseline strategies of coreference resolution: SINGLETONS and ONE-CLUSTER. These were explained in Section 3.7.

These baselines obviously correspond to very bad coreference resolution systems and, ideally, should be given low scores on an adequate evaluation metric.

It is clear that the B³ metric gives an unreasonably high score for the SINGLETONS baseline; e.g., the score is B³-F1 = 86.0 for German. In this case, the

-	CEAF			MUC		
Language	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
Baseline Singletons: each markable is an entity						
Catalan	61.2	61.2	61.2	0.0	0.0	0.0
German	75.5	75.5	75.5	0.0	0.0	0.0
English	71.2	71.2	71.2	0.0	0.0	0.0
Spanish	62.2	62.2	62.2	0.0	0.0	0.0
Italian	71.1	71.1	71.1	0.0	0.0	0.0
Dutch	34.5	34.5	34.5	0.0	0.0	0.0
Baseline AllInOne: all markables form one entity						
Catalan	11.8	11.8	11.8	100	39.3	56.4
German	8.2	8.2	8.2	100	24.8	39.7
English	10.5	10.5	10.5	100	29.2	45.2
Spanish	11.9	11.9	11.9	100	38.3	55.4
Italian	11.4	11.4	11.4	100	29.0	45.0
Dutch	19.7	19.7	19.7	100	66.3	79.8
-	B ³			BLANC		
Language	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
Baseline Singletons: each markable is an entity						
Catalan	61.2	100	75.9	50.0	48.7	49.3
German	75.5	100	86.0	50.0	49.4	49.7
English	71.2	100	83.2	50.0	49.2	49.6
Spanish	62.2	100	76.7	50.0	48.8	49.4
Italian	71.1	100	83.1	50.0	49.2	49.6
Dutch	34.5	100	51.3	50.0	46.7	48.3
Baseline AllInOne: all markables form one entity						
Catalan	100	4.0	7.7	50.0	1.3	2.6
German	100	2.4	4.7	50.0	0.6	1.1
English	100	3.5	6.7	50.0	0.8	1.6
Spanish	100	3.9	7.6	50.0	1.2	2.4
Italian	100	2.1	4.1	50.0	0.8	1.5
Dutch	100	8.0	14.9	50.0	3.2	6.2

Table 4.5: Scores of the baselines.

score $MUC-F1=0$ could indicate the system is low performance.

Similarly, $MUC-F1$ gives an unreasonably high score for ONE-CLUSTER; e.g., the score is $MUC-F1 = 79.8$ for Dutch. In this case, the scores $B^3-F1=14.9$ and $CEAF-M-F1=19.7$ shows the low performance of the system.

Roughly speaking, the system tendency to incorrectly generate larger clusters is penalized in B^3 and $CEAF-M$ metrics, and to incorrectly generate singleton clusters is penalized in MUC metric. For example, from the MUC , B^3 and $CEAF-M$ scores of the SINGLETONES and ONE-CLUSTER baselines, it can be predicted that the Dutch data set has many fewer singletons (a considerably higher MUC score for the ONE-CLUSTER baseline) than German and Italian data sets (a considerably higher B^3 and $CEAF-M$ score for the SINGLETONES baseline).

Since at the SemEval-2010 there was no agreement on a standard measure for coreference resolution evaluation, the task organizers were not able to declare a winner. However, a simple approach could have been to average all scores of one system and select the system with the best average as the winner. In this case, SUCRE would have been declared the winner.

4.8 CoNLL-2011

SUCRE successfully participated in CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNote [Pradhan et al., 2011b]. The same system was used as in SemEval-2010. In the following section the task will be briefly described and the results reported.

4.8.1 Task

The CoNLL-2011 shared task used the English language portion of the OntoNotes data. It consists of more than one million words from newswire (450K), magazine articles (150K), broadcast news (200K), broadcast conversations (200K) and web data (200K). The task was coreference resolution of entities and events given predicted information on the other layers. Pradhan et al. [2007a] did preliminary investigations using this data. To cover multiple genres of OntoNotes coreference data, a test set spanning all the genres was used for evaluation.

To prevent the winner declaration problem (see 4.7.6), the CoNLL-2011 organizers introduced a single official score to determine the winning system. This official score is the unweighted average of MUC, B³ and CEAF_E scores:

$$\text{CoNLL-2011 Official Score} = \frac{(MUC + B^3 + CEAF_E)}{3}$$

4.8.2 Data set

OntoNotes was used for the CoNLL-2011 shared task. The OntoNotes project¹ is to provide a large-scale, accurate corpus for general anaphoric coreference. It aims to cover entities and events (i.e. it is not limited to noun phrases or a limited set of entity types) [Pradhan et al., 2007a].

For training we used 4,674 documents containing a total of 1,909,175 tokens, 190,700 markables and 50,612 chains.

SUCRE participated in the closed track of the shared task. Experiments were performed for the two kind of documents, namely, the automatically preprocessed documents and the gold preprocessed documents. I report the scores on the development data set using the official scorer of the shared task. The automatically preprocessed part consists of 303 documents containing a total of 136,257 tokens, 52,189 automatically detected markables, 14,291 true markables and 3,752 chains. The gold preprocessed part consists of 303 documents containing a total of 136,257 tokens, 52,262 automatically detected markables, 13,789 true markables and 3,752 chains.

4.8.3 Scorer Problem

The CoNLL-2011 shared task had some evaluation problems. One of the problems was a bug in *the official scorer* which was found after the participants had submitted their final results. After the first evaluation by the task organizers (see figure 4.6) with the bugged scorer, SUCRE came in first place among 21 participants. After that they found the bug and fixed it (see figure 4.7), SUCRE went down to tenth place. From a competition point of view, a change in the official scorer used for system tuning during development was not fair. But, as we believe

¹<http://www.bbn.com/ontonotes/>

	Rec.	Prec.	F ₁
MUC	84.17	61.63	71.16
B ³	88.45	63.00	73.59
CEAF _M	42.70	37.20	39.76
CEAF _E	31.14	34.09	32.55
BLANC	61.86	63.51	62.61
OFFICIAL SCORE: 59.10			

Table 4.6: Results of SUCRE on the test data set for the automatically detected markables evaluated by the bugged scorer. The score of the best system is 57.79 [Pradhan et al., 2011a].

	Rec.	Prec.	F ₁
MUC	55.64	51.50	53.49
B ³	69.66	62.43	65.85
CEAF _M	42.70	42.70	42.70
CEAF _E	32.33	35.40	33.79
BLANC	61.86	63.51	62.61
OFFICIAL SCORE: 51.04			

Table 4.7: Results of SUCRE on the test data set for the automatically detected markables evaluated by the fixed scorer.

	Automatic			Gold		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
MD	60.17	60.92	60.55	62.50	61.62	62.06
MUC	54.30	51.84	53.06	57.44	53.15	55.21
B ³	71.39	64.68	67.87	74.07	64.39	68.89
CEAF _M	46.36	46.36	46.36	47.07	47.07	47.07
CEAF _E	35.38	37.26	35.30	35.19	38.44	36.74
BLANC	65.01	64.93	64.97	66.23	65.16	65.67

Table 4.8: Results of SUCRE on the development data set for the automatically detected markables. MD: Markable Detection.

that such shared tasks are to promote awareness and to establish pathways for further research, we are happy that the organizers found the bug and fixed it, despite the fact that our system fell from first to tenth place.

4.8.4 Results

Here recall, precision, and F_1 for MUC [Vilain et al., 1995], B³ [Bagga and Baldwin, 1998], CEAF_M/CEAF_E [Luo, 2005] and BLANC [Recasens et al., 2010] will be reported on the development data set.

Table 4.8 presents the results of our system for the automatically detected markables. It is apparent from this table that the application of the gold pre-processed documents slightly improves performance (MD-F1: +1.51; MUC-F1: +2.15; B³-F1: +1.02; CEAF_M-F1: +0.71; CEAF_E-F1: +1.44; BLANC-F1: +0.70).

Table 4.9 presents the results of our system for only those true markables that were part of coreference chains. Again the results show that the application of gold preprocessed documents slightly improves the performance (MUC-F1: +1.44; B³-F1: +0.86; CEAF_M-F1: +1.1; CEAF_E-F1: +1.26; BLANC-F1: +0.37).

Comparing the results of tables 4.8 and 4.9, there is a significant difference between the scores on the automatically detected markables and the scores on the true markables (e.g. for the automatically preprocessed documents: MUC-

	Automatic			Gold		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
MUC	58.63	87.88	70.34	60.48	88.25	71.78
B ³	57.91	86.47	69.36	59.21	86.25	70.22
CEAF _M	59.81	59.81	59.81	60.91	60.91	60.91
CEAF _E	70.49	36.43	48.04	71.09	37.73	49.30
BLANC	69.67	76.27	72.34	70.34	76.01	72.71

Table 4.9: Results of SUCRE on the development data set for the true markables (i.e. no singleton is included).

F1: +17.28; CEAF_M-F1: +13.45; CEAF_E-F1: +12.74; BLANC-F1: +7.37). No significant improvement in B³ can be determined (automatic: +1.49; gold: +1.33). I suspect that this is partly due to the very sensitive nature of B³ against the singleton chains. This is because in the implementation of scorer for the CoNLL-2011 shared task, the non-detected key markables are automatically included in the response as singletons.

4.9 Conclusion

In this chapter, we presented SUCRE, a modular supervised system for coreference resolution. In comparison with the existing systems, the most important advantage of our system is its modularity and its flexible method of feature engineering.

There are four classifiers integrated in SUCRE: Decision-Tree, Naive-Bayes, SVM and Maximum-Entropy. The system uses thresholded best-first clustering. It searches for the best predicted antecedent from right-to-left starting from the end of the document in the range above the threshold.

SUCRE successfully participated in SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010] for gold and regular closed annotation tracks of six languages. It obtained the best results in several categories, including the regular closed annotation tracks of English, German and Italian.

SUCRE also participated in CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNote [Pradhan et al., 2011b]. We showed that the application of the gold preprocessed documents improves performance. It has also been demonstrated that the availability of the true markables significantly improves the results.

Chapter 5

Relational Feature Engineering

5.1 Introduction

In this chapter, a new framework for feature engineering of natural language processing (NLP) will be presented that is based on representing unstructured text in a relational data model. It includes powerful and flexible methods for implementing and extracting new features and thereby reduces the time and effort needed for creating a natural language processing system.

Relational data modeling is a well-known technology for structured data modeling and is supported by a wide array of software and tools. Converting a text corpus to/from its equivalent relational database model is straightforward in our framework.

The aim is to generalize previous approaches that use the relational model for NLP to create a framework that can represent more complex linguistic structures and support more complex feature definitions. At its core is a novel systematic approach for defining features of the textual and attributive entities in NLP.

The framework outlined here is intended to support the process of designing and realizing a machine-learning based NLP system in a number of ways:

- (i) At the heart of the framework is the relational data model, which provides a common structure in a uniform and well-understood format for all the data that are used at any stage during the process: the text corpus, pre-processing results, relations between entities, feature definitions, feature values, classification results, etc. As a result, there is more coherence of

the methods and algorithms; and implementation, maintenance, reuse of components and integration of external components are all facilitated.

- (ii) The relational model allows the usage of SQL as the formalism for defining and extracting features. SQL is fast and powerful. It supports a rapid development cycle in which a feature can be defined, tested and modified in a few minutes. This greatly reduces the effort of creating an NLP system for a particular task.
- (iii) The framework also makes it easy for the developer to reduce the number of features in cases where this is desirable. Removing redundant features often improves classification accuracy. Smaller feature sets can also be more robust and cut down classification times. If the latter is the main motivation for removing features, the developer can rapidly remove those features that have the smallest effect on classification accuracy.
- (iv) The framework's modular architecture provides a clean separation between data storage, feature engineering and machine learning algorithms. As a result, it is easy to integrate any additional machine learning algorithm into the system.
- (v) Many algorithms from the knowledge discovery and data mining community have been implemented for relational databases. Since a relational model is being used for all the data here, it is possible to use these methods for NLP with little design and implementation effort.

To demonstrate the utility of our framework in NLP feature engineering, an instantiation and evaluation of the framework for the problem of coreference resolution in multiple languages will be presented. Our system SUCRE, built with the framework, achieved competitive results in a short implementation period doing a full coreference resolution of named entities, pronouns, and full noun phrases on the data sets of the six languages from SemEval-2010 Task 1 Coreference Resolution in Multiple Languages, the results of which were presented in Chapter 4.

This chapter is organized as follows. Section 5.2 presents related work. In Section 5.3, the process model of the framework will be described. The data

model is conceptualized in Section 5.4. Section 5.5 describes our approach to feature engineering. The Framework Architecture is introduced in Section 5.6. In Section 5.7, an instantiation of our framework for coreference resolution will be presented. Section 5.8 presents feature engineering of coreference resolution in multiple languages. Section 5.9 concludes this chapter.

5.2 Related Work

Application of relational models to facilitate text processing was first proposed by Crawford and Macleod [1978], where it was shown that SEQUEL (the first version of SQL) could be used for keyword search. Searching in relational textual databases and user-defined operators to manage text were discussed by Lynch and Stonebraker [1988]. Grossman introduced a database design for document retrieval within the relational model [Grossman, 1992]. He also uses relational databases for keyword-based information retrieval [Grossman et al., 1997]. I am not aware of other work that uses the relational model for fast, interactive feature engineering in NLP.

WEKA [Hall et al., 2009], a well-known data mining software, has a SQL viewer user interface that allows users to extract data from a database. But WEKA does not support the rapid iterative feature engineering that is the main purpose of the SUCRE framework; for example, it does not offer an interactive mechanism to define features that can seamlessly access relational data. Our framework, though, could be easily integrated into WEKA for feature definition and extraction for NLP tasks. An example of such an adaptation for geographic data processing has been proposed by Bogorny et al. [2006], where an interoperable module is presented that uses a Geographic Database Management System to model distance and topological spatial relationships.

5.3 Process Model

We are interested in solving NLP problems using statistical classifiers trained in a supervised setting. The design and realization of a system in this approach involves several steps, which will be illustrated below using the problem of BIO named entity recognition (NER) – the problem of identifying each word as the

initial word (B) of a named entity (NE), a non-initial part (I) of an NE or as not being part (O) of an NE.

1. **Define classification problem.** For NER, words are instances to be classified, and there are three classes: B, I and O. Below, instances will be referred to as *samples*.
2. **Provide training set.** In NER, this is often text in which each word has been annotated as B, I or O. The training set must conform to the real-world use of the function (e.g. a real-world named entity is: *The United Nations*)
3. **Define evaluation measure.** For NER, this could be a word-level measure or an NE-level measure (precision, recall, F_1 of NEs recognized)
4. **Select learning method and classifier.** (e.g., decision tree, multinomial Naive Bayes, tree-kernel SVM etc)
5. **Develop feature representation for instances.** This is usually done through many iterations, in each of which the current feature representation is evaluated and analyzed on a validation set and then modified. This step is often the most difficult and time consuming, but it is critical for building a successful system. To a large extent, the accuracy of classification in NLP depends on the quality of the features.
6. **Parameter tuning.** Parameters of the learning algorithm may be tuned (on a validation set or by cross-validation)
7. **Final test.** Test of performance of complete system on a held-out test set

Taking the above steps into account, the advantages of our framework are as follows:

- More coherency of the methods and algorithms that are needed in different stages by presenting a common structure of the text corpus based on the relations between the textual entities.
- A natural formalism for supporting the definition and extraction of features.

- It can be employed for reducing the size of the feature vector (in the feature space of an NLP task).
- Its modular approach makes it possible to use it with other machine learning tools.
- It is possible to apply many existing methods of knowledge discovery in the database to the text corpus with less design and implementation efforts.

5.4 Data Model

Relational data modeling is a well-known method for structured data and is supported by a wide range of software and many tools. The main purpose of the relational model in our framework is to enable the use of a language like SQL [Connolly and Begg, 1998] for feature definition. Using SQL, originally based on relational algebra (an offshoot of first-order logic), as the feature definition language is a method that is both easy to use and flexible when extracting different features from the relational data model of the text corpus.

The relational data model is the basis for defining the underlying structures of the text. To the extent that features in natural language processing are based on the values of attributes of textual entities and on relations between entities, the relational database model is a natural formalism for supporting the definition and extraction of features. Converting a text corpus to its equivalent relational data model in a preprocessing step is simple and efficient, as will be shown below.

The main relation (table)¹ of the data model is built of tokens. This relation has one tuple (row) per token indexed from the beginning of the corpus. This tuple contains the token itself and its attributes (e.g. part-of-speech and lemma). According to this main relation (which is enough to rebuild a text corpus) other relations are defined based on the underlying structures and their attributes. These relations model text structures such as documents, paragraphs, sentences, noun phrases and named entities and are defined on top of the token relation. For example, each tuple of the noun phrase relation contains, in addition to its attribute values, the indexes of its first, last and head words. All these relations

¹We use the formal terms *relation*, *tuple* and *attribute* for *table*, *row* and *column*.

and the corresponding attributes are used to define some new attributes in order to finally arrive at a desired set of attributes that act as the input feature vector to the employed learning algorithm.

The basic relations are defined below:

- **Token:**
(Token-ID, Sentence-ID, Token, Attribute-Vector)
- **Sentence:**
(Sentence-ID, Paragraph-ID, Attribute-Vector)
- **Paragraph:**
(Paragraph-ID, Document-ID, Attribute-Vector)
- **Document:**
(Document-ID, Attribute-Vector)

For each specific NLP task, new task-specific relations are needed. For example, for NER, a new relation NounPhrase would be defined as follows:

- **NounPhrase:**
(NounPhrase-ID, First-Token-ID, Last-Token-ID, Head-Token-ID, Attribute-Vector)

It is evident that SQL can be used to manage the text data. For example, with the following SQL statement we can access the words of a noun phrase X:

```
SELECT * from Token
INNER JOIN NounPhrase
ON NounPhrase-ID = X AND
   (Token-ID >= First-Token-ID AND
    Token-ID <= Last-Token-ID)
```

After defining the basic and task-specific relations for a particular NLP task, four relations for learning and classification (concerning samples and their corresponding feature vectors) are needed, such as:

- **TrainSamples:**
(Sample-ID, {Relational Connections}, Class)

- **TrainSamplesFeatureVector:**
(Sample-ID, Feature-Vector)
- **TestSamples:**
(Sample-ID, {Relational Connections}, Estimated Class)
- **TestSamplesFeatureVector:**
(Sample-ID, Feature-Vector)

In **TrainSamples** and **TestSamples**, *Relational Connections* is the set of referential constraints (foreign keys) that must be designed for a particular NLP task. For example, if the training samples are a subset of the set of all noun phrases in the text, then *Relational Connections* would have one member, NounPhrase-ID, and the TrainSamples relation would look as follows:

- **TrainSamples:**
(Sample-ID, NounPhrase-ID, Class)

Or if the training samples are pairs of noun phrases, then *Relational Connections* would have two members, NounPhrase-1-ID and NounPhrase-2-ID, and the TrainSamples table would be as follows:

- **TrainSamples:**
(Sample-ID, NounPhrase-1-ID, NounPhrase-2-ID, Class)

Feature-Vector in **TrainSamplesFeatureVector** and **TestSamplesFeatureVector** is a vector of attributes that is defined based on the relations of the model. Each element of this vector is defined as an SQL statement that calculates its value. An example feature for the above mentioned TrainSamples would be: *the head word of the noun phrase is a proper noun*. The simplified formulation of this feature in SQL would be (sample Y refers to noun phrase X via NounPhrase-ID):

```
UPDATE TrainSamplesFeatureVector
SET Feature-Vector[1] =
SELECT * from Token
INNER JOIN NounPhrase
ON NounPhrase-ID = X AND
```

```
(Token-ID == Head-Token-ID AND  
Token.Attribute-Vector[1] == "NP")  
WHERE TrainSamplesFeatureVector.Sample-ID = Y
```

In the above SQL example, the token attribute 1 (`Token.Attribute-Vector[1]`) is part of speech and "NP" means proper noun.

SQL supports the definition of more complex relations and attributes than the ones given above, as will be demonstrated in our description of the example system for coreference resolution in section 5.7.

5.5 Relational Feature Engineering

SQL includes operators on relations that are used to define features, including arithmetic, concatenation, comparison, logical, set and user-defined operators. Each feature definition is an attribute (column) of a relation (the feature vector table). In our setup, the feature extraction process consists of the calculation of features and their storage in the feature vector table.

The key capabilities of the relational model that we exploit are (i) all data is available in a uniform relational model, and (ii) SQL operators provide a uniform and powerful way of manipulating the data, in particular, for defining features. As a result, the developer can easily and quickly define features, thereby exploring the feature space.

In addition to SQL operators, we can define our own functions such as the edit distance calculator or the feature binarizer. In binarization, features with numerical values are converted into binary features by finding an optimal threshold value. For example, for the attribute distance between two noun phrases, the system finds the threshold θ that gives the best performance for the binary feature "distance is greater than θ ". Performance of a feature can be measured by a number of different figures of merit, such as information gain ratio (see below).

In addition to features that are directly defined and computed on the relations, it is also possible to import externally-generated features. For this purpose, it is enough to relate an externally-generated feature value to its corresponding textual entity. This approach to feature engineering is suitable when a complex preprocessing pipeline provides complex features for the NLP task. However, our

method tries to include all basic kinds of atomic and pair features, but there are still a few cases that our method cannot cover. For such cases the system accepts externally-extracted features. For example, a semantic similarity measure that is calculated using a search engine can be imported as an external feature and be combined with any other available feature in the system.

After these initial definition, the features are engineered with the goal of creating a feature set that will result in good performance. Feature engineering, as a process, consists of two parts:

- Feature definition: done using a feature definition language (SQL or pseudo-language)
- Feature selection: there are two models of feature selection [Liu and Motoda, 1998] that can be considered: the *filter model* and the *wrapper model*. In the filter model, there is a set of features, and the final feature set is the subset that gives the best learning result. In the wrapper approach, a method for semiautomatic exploration of the feature space is used to find better features. Our method of feature engineering can be categorized as a wrapper model, where it is possible to search the feature space by defining new features or combining them to find the best possible feature set according to the result of training and evaluation data sets.

Feature definition and selection in our setup is an iterative process that consists of the following steps:

1. Definition of new features (new definition from scratch or definition by combining existing features)
2. Evaluation of the features using the figure of merit
3. Inclusion of promising features in the current feature set and removal of non-performing features
4. Data analysis to determine which phenomena are not captured by the current feature set

Clearly, *a relational data model* does not mean that *a relational database management system* (RDBMS) is always needed to manage the data. For example,

for online processing of only one document with 10 sentences, an RDBMS (i.e., importing and processing in RDBMS) could be too expensive. In such cases this relational data model can be designed and implemented using any desired programming language. The important point is the representation of the text in the relational model and the ease with which this simple text format can be used to model the structure of the text to extract the features.

Figure of merit for features In principle, we could use any figure of merit to evaluate the usefulness of a feature or to compare two similar features, including Gini index (Equation 2.16) and information gain (Equation 2.18). In our current system, expected information gain (IG, see Equation 2.18) and information gain ratio (IGR, see Equation 2.21) are used.

As an example, consider the following two features, which can be considered different attempts to formalize the same linguistic property:

1. The noun phrase has a subject role and is *definite* (e.g. its first word is a *definite* article)
2. The noun phrase has a subject role and is *not indefinite* (e.g. its first word is not an *indefinite* article)

The information gain ratios of the above mentioned features are equal to 0.0026 for the first and 0.0051 for the second – this shows that the second one is the better choice. It implies that IGR can be used as an indicator for which features are likely to improve classification accuracy.

5.6 Framework Architecture

The architecture of the system has two main parts:

1. **Data Modeling**

In a typical NLP system this part is known as preprocessing. Here the text corpus is processed and converted to a relational data model. The main functionalities in this stage are:

- (a) Preliminary text conversion (e.g. tokenization)

- (b) Extracting atomic attributes of tokens and basic structures (sentences, paragraphs and documents)
- (c) Task-specific structure detection (e.g. noun phrases)
- (d) Extracting atomic attributes of task specific structures (e.g. syntactical role of noun phrases)

2. Classification

After a relational modeling of the text corpus, classification can be done. Its components are:

- (a) Instance generation: can be interpreted as the sampling step in statistical learning.
- (b) Feature definition and extraction: here the feature definition and engineering methods which were described before are applied.
- (c) Learning (training): any machine learning method that accepts the feature representation defined in the previous step can be used.
- (d) Classification (test): the classifier learned by the machine learning method is applied.

In the following section, an instantiation and evaluation of this framework for the problem of coreference resolution in multiple languages will be presented.

5.7 Coreference Resolution

Considerable engineering effort is needed to build a full coreference resolution system, and a major part of this effort concerns the feature engineering. Therefore, a framework that is able to define and extract the features based on a feature definition language can help the researcher save a lot of implementation effort needed for feature definition and extraction.

We report here on the design and realization of a machine-learning based coreference resolution system in the steps outlined before. We start with a high-level description of the instantiations of the steps that were undertaken for building the system before describing architecture and results in more detail.

1. **Define classification problem.** As stated above, we used one of the standard models for statistical coreference resolution: instances are ordered pairs of markables (e.g., the pair $\langle \textit{Bill Clinton}, \textit{he} \rangle$) and the two classes are *coreferent* (the two markables refer to the same entity) and *disreferent* (the two markables refer to different entities).
2. **Provide training set.** We used the training sets provided by SemEval-2010.
3. **Define evaluation measure.** As stated above, markables are partitioned based on the classification decisions on markable pairs. The partitioning was then evaluated using the SemEval-2010 measures MUC [Vilain et al., 1995], B³ [Bagga and Baldwin, 1998], CEAF [Luo, 2005] and BLANC [Recasens and Hovy, 2010].
4. **Select learning method and classifier.** We used the following four: decision tree, Naive Bayes, support vector machine, maximum entropy classifier.
5. **Develop feature representation for instances.** We started with six feature sets (one for each language), which were derived from a feature set for English that contains standard baseline features used in coreference resolution, similar to the sets used by Soon et al. [2001] and Ng and Cardie [2002]. We then went through several iterations for each of the six languages before settling on the final feature sets.
6. **Parameter tuning.**
7. **Final test.** Results on the test set published by SemEval-2010 were submitted to the SemEval-2010 competition and evaluated by the SemEval organizers.

There are some available systems that perform coreference resolution, such as BART [Versley et al., 2008b], GUITAR [Steinberger et al., 2007] and JAVARAP [Qiu et al., 2004]. None of these systems offer a way of defining, searching and selecting features for coreference resolution that is as flexible and efficient as the one offered in SUCRE. For example, BART feature extractors are realized

as separate classes, allowing for their independent development [Versley et al., 2008b]. That means that for each new feature set a new class must be developed and added to BART. In contrast to SUCRE, no rapid cycle of feature definition, combination and testing is supported.

5.7.1 System Description

Conforming to the framework architecture, the system has two main parts: data modeling and classification. These two parts will have been described in detail for the coreference resolution application.

1. Data Modeling (Preprocessing):

(a) Preliminary text conversion (e.g. tokenization)

(b) Extracting atomic attributes:

For tokens:

- Attribute 1: part-of-speech tag (e.g. proper noun, verb, article)
- Attribute 2: grammatical gender (male, female or neuter)
- Attribute 3: natural gender (male or female)
- Attribute 4: number (e.g. single, plural or both)
- Attribute 5: semantic class (e.g. person, location, time)
- Attribute 6: pronoun type (e.g. personal, reflexive, demonstrative)
- Attribute 7: case (e.g. nominative, accusative, dative or genitive in German)
- Attribute 8: person pronoun (first, second or third)

For sentences:

- Attribute 1: sentence type (e.g. simple, compound or complex)
- Attribute 2: number of words

For paragraph:

- Attribute 1: number of words
- Attribute 2: number of sentences

For document:

- Attribute 1: document type (e.g. news, article, book and ...)
 - Attribute 2: number of sentences
 - Attribute 3: number of paragraphs
- (c) Markable detection: all noun phrases from the parse information.
- (d) Extracting atomic markable attributes:
- Attribute 1: semantic class
 - Attribute 2: syntactical role

2. Classification (Coreference Resolution):

- (a) Pair generation
- (b) Pair feature definition and extraction (i.e. a feature vector for each markable pair)
- (c) Learning
- (d) Classification and Partitioning

Relational Data Model of Text Corpus for Coreference Resolution

We will first look at the feature concept of coreference resolution that motivates the particular database design we chose for coreference resolution.

In the approach to coreference resolution we have chosen, features are **pair features** (see 3.2.2). For pair feature definition and extraction, the head words of markables are usually used, but in some cases the head word is not a suitable choice. For example, consider these two markables: *the university students in Germany* and *the university students in France*. In this case the head words and the first four words of each markable are the same but they cannot be coreferent, and this can be seen only by looking at the entire noun phrase. Some features require complex preprocessing or complex SQL definitions. Consider the two markables *the members of parliament* and *the members of the European Union*. The semantic class of *members* is *person* in the first case but *country* in the second. To cover all such cases, the SQL feature definitions must be able to

access all information that is connected to a markable, including the first, last and head words of the two markables, all other words of the two markables, and the two markables as atomic elements.

Tables 5.1 and 5.2 present the relational data model of the text corpus that satisfies these desiderata. It has a structure that is easy to generate.

In the token table, Token-ID is the position of token counting from the beginning of the corpus. It is used as the primary key to uniquely identify each token. Sentence-ID is counting from the first of the corpus and at the same time is the foreign key pointing to the primary key of sentence table. Atomic token attributes can be stored in the columns of the token table called Token-Attribute-X. It is obvious that the plain text, along with any other format of the corpus, can be generated from the token table.

In the markable table, Markable-ID is the primary key. First-Token-ID, Last-Token-ID and Head-Token-ID refer to the token table. In the pair table, Pair-ID is the primary key; Markable-1-ID and Markable-2-ID refer to the markable table.

As was shown before, the features can be directly defined in SQL, but this would make the system cumbersome for the average NLP developer. Therefore, we have also designed a pseudo-language which is syntactically more convenient. In particular, it contains keywords for the entities we need to refer to. These keywords are presented in the *Abbreviation* column of Tables 5.1 and 5.2. Roughly speaking, the pseudo-language embeds all the needed inner joins to provide fluent access to the attributes of different relations.

Relational Feature Definition Language

We call our feature definition language *RFDL* (Relational Feature Definition Language), for which we designed a formal grammar $G_{RFDL}(V, T, P, S)$, where:

V is the set of symbols called variables.

S is the starting or goal variable from **V**.

T is the set of symbols called terminal.

P is the set of productions.

Figure 5.3 shows the set of variables, along with a short description.

Figure 5.4 presents the set of terminals **T**.

Column	Characteristic	Abbreviation
Token Table		tt
Token-ID	Primary Key	tid
Sentence-ID	Foreign Key	stc
Word-String	Attribute	str
Attribute-1	Attribute	a1
Attribute-2	Attribute	a2
...	Attribute	...
Attribute-8	Attribute	a8
Sentence Table		st
Sentence-ID	Primary Key	sid
Paragraph-ID	Foreign Key	prf
Attribute-1	Attribute	a1
Attribute-2	Attribute	a2
Paragraph Table		pt
Paragraph-ID	Primary Key	pid
Document-ID	Foreign Key	doc
Attribute-1	Attribute	a1
Attribute-2	Attribute	a2
Document Table		dt
Document-ID	Primary Key	did
Attribute-1	Attribute	a1
Attribute-2	Attribute	a2
Attribute-3	Attribute	a3
Markable Table		mt
Markable-ID	Primary Key	mid
First-Token-ID	Foreign Key	ft
Last-Token-ID	Foreign Key	lt
Head-Token-ID	Foreign Key	ht
Attribute-1	Attribute	a1
Attribute-2	Attribute	a2

Table 5.1: Relational Data Model of Text Corpus for Coreference Resolution

Table of Training Pairs		ttp
Pair-ID	Primary Key	pid
Markable-1-ID	Foreign Key	m1
Markable-2-ID	Foreign Key	m2
Class	Attribute	cls

Table of Training Feature Vectors		tffv
Pair-ID	Primary Key	pid
Feature-1	Attribute	f1
Feature-2	Attribute	f2
...	Attribute	...
Feature-N	Attribute	fn

Table of Test Pairs (to be estimated pairs)		tep
Pair-ID	Primary Key	pid
Markable-1-ID	Foreign Key	m1
Markable-2-ID	Foreign Key	m2
Estimated Class	Attribute	estcls

Table of Test Feature Vectors		tefv
Pair-ID	Primary Key	pid
Feature-1	Attribute	f1
Feature-2	Attribute	f2
...	Attribute	...
Feature-N	Attribute	fn

Table 5.2: Relational Data Model of Text Corpus for Coreference Resolution

Variable	Description
S	Start
I	Identifier
R	String
A	Attribute Value (e.g. “a6.reflexive”)
O	Operator (e.g. “+”, “/” and “==”)
F	Function (e.g. “min”, “max” and “abs”)
N	Number (e.g. “90” and “237”)
M	Markable (e.g. “m1” and “m2”)
W	Word in Markable (e.g. “ft”, “ht” and “lt”)
E	Markable Variable (e.g. “m1.ft.stc”)
T	Pair Tables of Data Model (e.g. “ttp” and “tep”)
L	Feature Vector Tables of Data Model (e.g. “ttfv” and “tefv”)

Table 5.3: Variable set V

Terminals
“=”, “-”, “+”, “*”, “/”, “==”, “!=”, “<”, “>”, “<=”, “>=”, “ ”, “&&”, “(”, “)”, “{”, “}”, “.”, “[”, “]”, “a ... z”, “A ... Z”, “0 ... 9”, “abs”, “extlf”, “ptreepairrel”, “ptreem1path”, “ptreem2path”, “max”, “min”, “dist”, “strmatch”, “seqmatch”, “strmatchlc”, “seqmatchlc”, “bswitch”, “eswitch”, “bswitchlc”, “eswitchlc”, “editdist”, “alias”, “minfo”, “m1”, “m2”, “ft”, “ht”, “lt”, “str”, “stc”, “ptl”, “fv1”, “ptc”, “fvc”, “pid”, “cls”, “estcls” “tid”, “sid”, “pid”, “did”, “mid”

Table 5.4: Terminal set T

Production Rules
$S \rightarrow L.fN=I$
$I \rightarrow N A E (I) IOI F(I) F(I,I)$
$N \rightarrow NN 0 1 2 \dots 9$
$A \rightarrow aN.R$
$R \rightarrow RR a b \dots z A B \dots Z 0 1 \dots 9$
$O \rightarrow - + * / == != < > <= >= \&\&$
$E \rightarrow T.M.aN T.M.W.aN T.M.W.stc T.M.W.str$
$M \rightarrow m1 m2$
$W \rightarrow ft ht lt$
$L \rightarrow ttfv tefv$
$T \rightarrow ttp tep$
$F \rightarrow abs extlf max min dist strmatch$
seqmatch strmatchlc seqmatchlc bswitch eswitch
bswitchlc eswitchlc editdist alias minfo

Table 5.5: Grammar

Examples

- `ttfv.f1 =`
 `(ttp.m1.ft.stc == ttp.m2.ft.stc) &&`
 `(ttp.m2.ht.a6 == reflexive)`

Feature 1 is set to 1 if two markables are in the same sentence and the type of the second markable head word is reflexive (a6 is pronoun type in SUCRE); otherwise, it is set to 0.

- `ttfv.f2 =`
 `(ttp.m1.ht.str == ttp.m2.ht.str) &&`
 `(ttp.m1.ht.a1 == a1.N) &&`
 `(ttp.m2.ht.a1 == a1.N)`

Feature 2 is set to 1 if the head words of the two markables exactly match and are nouns (a1 is part of speech in SUCRE); otherwise, it is set to 0.

- `ttfv.f3 =`
 `abs(ttp.m2.ft.stc - ttp.m1.ft.stc)`

Feature 3 is the distance (in sentences) between the two markables.

- `ttfv.f4 =`
 `(ttp.m1.ht.a1 == a1.N) &&`
 `(ttp.m2.ft.a1 == a1.P)`

Feature 4 is set to 1 if the head word of the first markable is a noun and the beginning word of the second markable is a pronoun; otherwise, it is set to 0.

- `ttfv.f5 =`
 `(ttp.m1.a1 == ttp.m2.a1)`

Feature 5 is set to 1 if both markables have the same semantic class. (a1 is semantic class of markable in SUCRE); otherwise, it is set to 0.

It is possible that two or more features build a new feature. For example, in the above mentioned features the conjunction of the first and the last one can be a new feature:

- `ttfv.f6 =`
 `(ttp.m1.ft.stc == ttp.m2.ft.stc) &&`
 `(ttp.m2.ht.a6 == reflexive) &&`
 `(ttp.m1.a1 == ttp.m2.a1)`

Feature 6 is set to 1 if two markables are in the same sentence and the type of the second markable head word is reflexive and both markables have the same semantic class; otherwise, it is set to 0.

In this way, we are able to explore the feature set until more optimized features are found or until the number of features has been reduced.

In addition to the feature functions that define the feature representation for statistical classification, we also defined **filters** which were described in 4.4 (Filters are high-accuracy pair features which can be used directly to filter the markable pairs while generating pairs or partitioning).

The filter feature definition is identical to the pair feature definition. If a filter feature is true, no pair will be generated for it. Some examples of filter features are as follows:

- `((ttp.m1.ht.a2 == male) &&`
 `(ttp.m2.ht.a2 == female)) ||`
 `((ttp.m1.ht.a2 == female) &&`
 `(ttp.m2.ht.a2 == male))`

The genders of the head words of the two markables don't match (only for male and female cases, i.e. the neutral case will not be deleted). `a2` is the atomic token attribute-2 for grammatical gender (male, female, neuter).

- `(ttp.m1.ht.a4 != ttp.m2.ht.a4) &&`
 `(ttp.m1.ht.a4 != unknown)`

The grammatical number features of the head words of the two markables don't match and the number of the second markable's head word is known (not equal to unknown). `a4` is the atomic token attribute-4 for number (plural, singular, both, unknown).

5.7.2 Evaluation

The experiments which were described in Chapter 4 show that our method of feature engineering is state-of-the-art and should have general validity.

Of course, features can be extracted without feature definition languages. But as the number of possible combinations of attributes is very large, it would be very time-consuming to search for new combinations without a systematic optimized approach as is being proposed in this thesis. As a result, system development time would be much longer in a setting where a framework like SUCRE is not available.

To summarize, it is time-consuming and cumbersome to explore the space of all features for a particular NLP task. The optimized systematic approach used in our system makes this search easier and faster. The system developer will therefore be able to compare more features and to create a better-performing system.

5.8 CoRe Features in Multiple Languages

In this section, the aim is to shed light on the feature engineering of coreference resolution in multiple language. I investigate and report the relative contributions of the various feature types for coreference resolution of Dutch, German, Italian and Spanish. For this purpose, SUCRE will be used, as it provides a feature definition and extraction mechanisms that make it possible to uniformly define the various features for the different languages.

In recent years there has been substantial work on the problem of coreference resolution. Most methods have presented and reported on the benchmark data sets for English. The feature sets they used are based on [Soon et al., 2001].

I will try to define a common feature set that can be used for these languages (i.e. Dutch, German, Italian and Spanish). As with [Bengtson and Roth, 2008] a rather simple but state-of-the-art system was used to investigate and measure the relative contributions of the various features for coreference resolution in multiple languages.

In order to have a uniform feature definition mechanism for different languages, the above-described feature definition language was used, which provides

a uniform and powerful way of manipulating the data. In this way, we were able to compare the contributions of the various features.

5.8.1 Feature Engineering

A comprehensive set of features for four languages was defined based on previous coreference resolution systems for English, e.g. [Bengtson and Roth, 2008].

To study the various features across different languages three categories of features can be distinguished:

1. **Identical Features:** features whose concept and definition is identical for all languages. Examples of these features are: sentence distance between two markables, exact string matching, partial string matching, one markable spans the other.
2. **Universal Features:** features with the same concept but different realization. Examples of these features are: noun type features (e.g. if the first markable is definite or indefinite); pronoun type features (e.g. the second markable is a reflexive pronoun); grammatical role features (different combinations are dependent on the corresponding available roles in annotation); semantic class, number and gender agreement.
3. **Language Specific Features:** the features that can be defined specifically for a language (e.g. grammatical gender in German).

As the goal was to define a common feature set for different languages, here I discuss only identical and universal feature types, that is, features that are language independent.

As mentioned before, filter features are high-accuracy link features which can be used directly to filter the markable pairs while generating the links or partitioning. However, for the sake of less complexity the result of experiments according to their non-filter/filter feature types will not be reported on.

5.8.2 Data Sets

The following data sets were used. They were described in Chapter 4:

- KNACK-2002 corpus [Hoste and De Pauw, 2006].
- TüBa-D/Z treebank [Hinrichs et al., 2005].
- LiveMemories project [Rodriguez et al., 2010].
- AnCora-ES corpora [Recasens et al., 2010].

It is clear that the availability of markables unrealistically simplifies the coreference resolution task [Stoyanov et al., 2009]; however, as we are comparing the contribution of different features in the different languages, in order to have a comparable baseline, we used the gold-standard annotated markables of each data set.

5.8.3 Results

During the evaluation, and in particular for feature engineering, the test sets were not used. Results will be presented for three different evaluation metrics:

1. MUC [Vilain et al., 1995]
2. B³ [Bagga and Baldwin, 1998]
3. CEAF-M [Luo, 2005]

As was shown before, our system had the best MUC and BLANC scores for all languages at SemEval-2010 Task 1 Coreference Resolution in Multiple Languages [Recasens et al., 2010]. For the other metrics, if our system was not the best, the best results were only slightly better than ours. This is important because it shows that our cross-lingual comparison of feature utility is conducted within a state-of-the-art system and should have general validity.

Now the results of the experiments for the identical and universal features will be reported on.

Identical Features: for this category three main groups of features will be distinguished:

1. **String-based features:** previous work has shown that string-based features are essential in coreference resolution (e.g. [Ng and Cardie, 2002]). The following features can be defined:
 - (a) exact string matching of the head words of both markables.
 - (b) head word of first markable exactly matches any word in the second markable.
 - (c) head word of second markable exactly matches any word in the first markable.
 - (d) any word of one markable exactly matches any word of another one.
 - (e) sub-string matching of the head words of both markables.
 - (f) head word of first markable partially matches any word in the second markable.
 - (g) head word of second markable partially matches any word in the first markable.
 - (h) any word of one markable partially matches any word of another one.
 - (i) edit distance between the head words of the markables.
2. **Distance features:** Hobbs [1986] found that 98% of the pronoun antecedents are in the same or previous sentence as the pronoun. In another work based on UCREL Anaphoric Treebank corpus, McEnery et al. [1997] showed that in 85.64% of cases the antecedent is within a window of 3 sentences. To check this for different languages we defined these features:
 - (a) sentence distance between two markables.
 - (b) word distance between two markables.
3. **Span features:** here we define two features:
 - (a) one markable is completely in the span of the other one.
 - (b) one markable has overlap with the other one.

Table 5.6 presents the result for a system with identical features. For each language exactly the same feature definition set was used. From these results –

	MUC			B ³			CEAF-M		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
de	35.8	40.6	38.1	80.3	88.5	84.2	72.9	72.9	72.9
es	17.1	38.2	23.6	65.9	91.4	76.6	62.0	62.0	62.0
it	32.8	30.6	31.7	78.1	80.1	79.0	64.4	64.4	64.4
nl	33.7	41.9	37.1	72.8	79.9	76.2	59.3	59.3	59.3

Table 5.6: Results for the identical features (de: German, es: Spanish, it: Italian, nl: Dutch).

	MUC			B ³			CEAF-M		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
SINGLETONS baseline									
de	0.0	0.0	0.0	75.5	100	86.0	75.5	75.5	75.5
es	0.0	0.0	0.0	62.2	100	76.7	62.2	62.2	62.2
it	0.0	0.0	0.0	71.1	100	83.1	71.1	71.1	71.1
nl	0.0	0.0	0.0	34.5	100	51.3	34.5	34.5	34.5
ONE-CLUSTER baseline									
de	100	24.8	39.7	100	2.4	4.7	8.2	8.2	8.2
es	100	38.3	55.4	100	3.9	7.6	11.9	11.9	11.9
it	100	29.0	45.0	100	2.1	4.1	11.4	11.4	11.4
nl	100	66.3	79.8	100	8.0	14.9	19.7	19.7	19.7

Table 5.7: Scores of SINGLETONS and ONE-CLUSTER baselines

	MUC			B ³			CEAF-M		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
de	+18.9	+4.8	+11.6	+4.9	-7.4	-1.1	+0.5	+0.5	+0.5
es	+26.5	+5.8	+20.2	+7.2	-18.3	-3.5	-3.7	-3.7	-3.7
it	+9.3	+4.8	+6.8	+0.9	-4.6	-1.9	-0.7	-0.7	-0.7
nl	-3.2	+22.8	+4.4	-4.6	+16.2	+3.6	+9.8	+9.8	+9.8
av	+12.9	+9.5	+10.7	+2.1	-3.5	-0.7	+1.5	+1.5	+1.5

Table 5.8: Relative scores (compared to table 5.6) after adding the noun type features to the identical features (av: Average).

which are considerably higher than the lower limits of the corresponding ONE-CLUSTER and SINGLETONS baseline scores (table 5.7) – we can conclude that these link features *should be in the common feature set of the four languages*. I believe that these features can be used for building a simple pair-wise coreference resolution system of any language with minimum needed preprocessing.

Universal Features: The following main groups of universal features can be defined. In each group the feature definitions depend on the language, extracted information (e.g. parse tree) and tag sets (e.g. part of speech tags). As an example, consider this feature for German: *the markable is indefinite*, we can determine the indefiniteness of a markable (singular case) by looking at its first word and determining if it is a form of the indefinite article *ein*.

1. Noun type features:

The following features for each language were defined. In each case two features for the first and the second markable of the link were defined:

If the first/second markable in the link is

- (a) common noun
- (b) proper noun
- (c) definite
- (d) indefinite

	MUC			B^3			CEAF-M		
	Rec.	Prec.	F_1	Rec.	Prec.	F_1	Rec.	Prec.	F_1
de	+11.2	+2.7	+5.9	+2.2	-7.0	-2.8	-2.3	-2.3	-2.3
es	-3.8	+2.7	-0.8	-1.2	+8.8	+3.5	+4.3	+4.3	+4.3
it	+0.5	+1.6	+1.9	-2.6	0	-1.3	0	0	0
nl	+19.0	+6.5	+13.1	+8.0	-13.0	-3.0	-4.7	-4.7	-4.7
av	+6.7	+3.4	+5.0	+1.6	-2.8	-0.9	-0.7	-0.7	-0.7

Table 5.9: Relative scores (compared to table 5.6) after adding the pronoun type features to the identical features.

Table 5.8 shows the relative scores (compared to table 5.6) after adding noun type features to the identical features. The average of each column was computed in order to provide a comparison measure. The average relative F_1 -scores, which are presented in the last line of table 5.8 (MUC = +10.7, B^3 = -0.7, CEAF-M = +1.5), indicate that *noun type features should also be added to the common feature set of the four languages*. There is a clear improvement for MUC, a small improvement for CEAF-M and no consistent change for B^3 .

Due to the fact that the number of the singletons and clusters vary for the four data sets, improving all three metrics for all data sets with the same feature types is not always possible. Therefore, the substantial improvement of the F_1 -MUC score average (+10.7) in comparison with the small worsening of its corresponding F_1 - B^3 (-0.7) can be interpreted as follows: the system correctly builds larger clusters. Of course, by building larger clusters the number of singletons that incorrectly join a larger cluster increases; however, the F_1 - B^3 score indicates that there were only very few cases in which this happened.

2. Pronoun type features:

Two simple common features for all languages are “if the first markable in the link is pronoun” and “if the second markable in the link is pronoun”. Instead of these simple cases, it is possible to use the type of pronouns for feature definition. The above mentioned features could be replaced

with these features (in each case one feature for the first markable and one feature for the second markable):

If the first/second markable in the link is

- (a) first person pronoun
- (b) second person pronoun
- (c) third person pronoun

Table 5.9 shows the relative scores (compared to table 5.6) after adding the pronoun type features to the identical features. Despite defining different features for the different languages, from the result we can again conclude that *the pronoun type features should be added to the common feature set of the four languages*. Just as for the noun type features, the average of the relative F1-Score can be explained: there is the tendency to generate more larger correct clusters (MUC-F1 = +5.0) that would be penalized for the incorrect cases by B³ (F1 = -0.9) and CEAF-M (F1 = -0.7) scores.

I also measured the impact of using the fine-grained pronoun type features instead of simple features for German and Spanish data sets:

For German:

- (a) reflexive pronoun (e.g. *sich* ‘self’)
- (b) substituting possessive pronoun (e.g. *meins* ‘mine’)
- (c) attributive possessive pronoun (e.g. *mein* ‘my’)
- (d) relative pronoun (e.g. *der* ‘the’)
- (e) demonstrative pronoun (e.g. *dieser* ‘this’)
- (f) substituting indefinite pronoun (e.g. *keiner* ‘no’)
- (g) attributive indefinite pronoun (e.g. *beiden* ‘both’)

For Spanish:

- (a) possessive pronoun (e.g. *su* ‘his/her’)
- (b) demonstrative pronoun (e.g. *esta* ‘this’)

	MUC			B ³			CEAF-M		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
de	-1.8	+4.8	+1.6	-5.5	0	-2.2	-2.4	-2.4	-2.4
es	+18.8	+6.8	+16.4	+5.0	-10.9	-1.1	0	0	0
it	+11.0	+10.8	+10.9	+0.9	-4.4	-1.2	0	0	0
nl	+3.8	+3.6	+3.7	+1.2	0	+0.7	+1.9	+1.9	+1.9
av	+7.9	+6.5	+8.1	+0.4	-3.8	-0.9	-0.1	-0.1	-0.1

Table 5.10: Relative scores (compared to table 5.6) after adding the grammatical features to the identical features.

- (c) relative pronoun (e.g. *que* ‘which’)
- (d) personal pronoun (e.g. *le* ‘him/her’)

In both languages the overall results of the fine-grained pronoun type features were better. The relative F1-Scores for German are MUC=+2.8, B³=+0.9 and CEAF-M=+1.2, and for Spanish are MUC=+0.4, B³=+1.6 and CEAF-M=+2.1. Our interpretation of this result is that *a fine-grained annotation can improve the quality of coreference resolution.*

3. Grammatical features:

As in the case of pronoun type features, two simple features can be defined here: “the first markable is a subject” and “the second markable is a subject”. But from parse information, more features can be defined based on grammatical roles. The following six features for each of the four languages can also be defined:

If the first/second markable in the link is a

- (a) subject
- (b) direct object
- (c) indirect object

Table 5.10 presents the relative scores (compared to table 5.6) after adding the grammatical role features to the identical features. And again the result

	MUC			B ³			CEAF-M		
	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁	Rec.	Prec.	F ₁
de	+8.0	+3.4	+5.8	+2.0	-3.7	0	0	0	0
es	+13.3	+0.9	+10.5	+3.7	-11.3	-2.1	-3.8	-3.8	-3.8
it	+5.4	-2.5	+1.7	-1.6	+3.1	+0.6	+3.4	+3.4	+3.4
nl	+13.8	+7.8	+11.2	+4.3	-6.8	-1.1	-0.6	-0.6	-0.6
av	+10.1	+2.4	+7.3	+2.1	-4.7	-0.4	-0.2	-0.2	-0.2

Table 5.11: Relative scores (compared to table 5.6) after adding the agreement features to the identical features.

allows us to conclude that *the grammatical role features should also be added to the common feature set of the four languages.*

4. Agreement features:

We define these features:

If the markables in the link agree in

- (a) number
- (b) natural gender
- (c) semantic class

The impact of these features on performance depends on the quality of preprocessing (i.e. the quality of the annotation of the corpus). The relative scores (compared to table 5.6) of adding these features to the identical feature set are presented in Table 5.11. A final conclusion can therefore be made: *agreement features should also be in the common feature set of the four languages.*

The results presented in Table 4.3 are the final results of our system concerning all of the features described above.

We have defined a common feature set for the four languages. Roughly speaking *a common feature set for the four languages contains all the identical and universal features that were described and tested in this thesis.*

It should be noticed here that, using uniform annotation, most universal features could be dealt with, just as the identical features.

5.9 Conclusion

A new framework for feature engineering of natural language processing based on representing unstructured text in a relational database model has now been presented. It includes powerful and flexible methods for implementing and extracting new features. It allows systematic and fast search of the space of features and thereby reduces the time and effort needed for creating an NLP system. The relational database model at the heart of this framework makes it possible to use SQL as the feature definition language.

I presented an example application of this framework to the problem of coreference resolution. The results of the system in multiple languages show that our approach to feature engineering supports rapid and high-quality development of a machine-learning based NLP system.

Section 5.8 presented a comparative study of coreference resolution features across different languages. Important distinctions between *identical*, *universal* and *language-specific* features were introduced and investigated the first two groups for four languages, Dutch, German, Italian and Spanish.

I first showed that a system using identical features can serve as a robust baseline for coreference resolution systems.

I defined universal features as features that correspond to a universal linguistic concept that has different “implementations” in languages due to lexical and grammatical differences. I defined four broad classes of universal coreference resolution features: noun type features, pronoun type features, grammatical features, and agreement features. Our feature definition and computation mechanism makes it easy to uniformly define and test these feature types for different languages and compare their relative contributions.

I found that each of the four feature types increases performance compared to the identical-feature baseline. Our experiments also showed that a fine-grained annotation can improve the quality of coreference resolution.

This framework is as an important step towards tackling many of the tasks the NLP community is working on in a multilingual setting. Using a single system and

a unified framework for feature definition and classification is advantageous for meaningful cross-linguistic comparisons and for gaining insights into the problem that are not restricted to a single language.

Chapter 6

Unsupervised Coreference Resolution

6.1 Introduction

In this chapter, an unsupervised framework will be presented that bootstraps a complete coreference resolution (CoRe) system from *word associations* mined from a large unlabeled corpus.

There is great interest in solving natural language processing problems using statistical inference. As was discussed in chapter 2, features play an important role in this regard. To have a better understanding of the possible features, I propose a feature space model of natural language. This feature space model contains two feature spaces: shallow space with only one term-frequency-based feature; and rich space that is n-dimensional and contains linguistic features.

Using the shallow feature space, a case study consisting of a novel method for word alignment will be presented. This shallow feature space has also been used for a new unsupervised approach to coreference resolution (A-INF) that uses lexical association scores between heads of markables as the distance measure when clustering markables.

A novel unsupervised self-training method will also be presented that first labels a corpus using the unsupervised model A-INF and then trains the supervised model SUCRE on this automatically labeled training corpus. Even though training is accomplished on a labeled corpus, the labeling of the corpus is produced in

a completely automatic fashion, without recourse to human labeling. Thus, it is an unsupervised approach.

The main contributions of this chapter are as follows:

- It will be demonstrated that lexical information can be used to develop an unsupervised model for shallow coreference resolution (subsystem A-INF).
- An unsupervised self-trained method (UNSEL) will be introduced that takes a two-learner two-feature space approach. The two learners are A-INF and SUCRE. The feature spaces are the shallow and rich feature spaces.
- It will be shown that the performance of UNSEL is better than the performance of other unsupervised systems when it is self-trained on the automatically labeled corpus and uses the leveraging effect of rich linguistic features.
- Our framework is a flexible and modular framework that is able to learn from data with different quality and domain. Not only is it able to deal with shallow information spaces (A-INF), but it can also deliver competitive results for rich feature spaces (SUCRE and UNSEL).

This chapter is organized as follows. Feature spaces are introduced in section 6.2. In section 6.3, I present an application of our shallow feature space approach for word alignment. Section 6.4 describes a novel unsupervised method for coreference resolution. In section 6.5, our unsupervised self-trained model will be presented. Section 6.6 concludes this chapter.

6.2 Feature Spaces

Generally speaking, a feature space is an abstract n -dimensional space where each instance is represented as a point. The dimensionality of the space is equal to the number of features used to describe the instances. Similar instances are grouped together, which can be estimated using different approaches.

A more exact view of the possible feature spaces helps us to have a more exact definition of the problem. Therefore I propose a feature space model containing the following spaces.

6.2.1 Shallow Feature Space

In a 1-dimensional space there is only one feature. This feature is a similarity measure between the data objects which can be calculated by different term-frequency-based methods, like the Dice coefficient [Dice, 1945], the log-likelihood-ratio (LLR) or mutual information [Manning and Schütze, 1999].

The key benefit of this approach is that it is possible to calculate the features (e.g. mutual information between two words) from an unlabeled corpus in a completely unsupervised fashion. However, the shallow feature space has two serious weaknesses:

1. Its term-frequency-based approach fails to cover cases that do not occur (i.e. unseen data) in the unlabeled corpus (negatively affecting recall).
2. Its shallow approach fails to precisely find the fine-grained rules from the data set. In other words, it generalizes a potentially n-dimensional space into 1-dimensional space causing its power of discrimination to be reduced (negatively affecting precision).

The above mentioned weaknesses led to the expectation that the performance should be worse when compared to the n-dimensional feature space approaches. However its simplicity and the fact that it is completely unsupervised still make it attractive. Specifically it can be used for bootstrapping, which will be utilized to address the problem of coreference resolution in section 6.5.

This approach is not limited to the problem of coreference resolution. An application of this approach will be presented in section 6.3 for the problem of word alignment, which is a sub task of machine translation.

6.2.2 Linguistic Feature Space

Linguistic feature space is an n-dimensional space consisting of different types of features. As was discussed in chapter 2, an attribute is a property or characteristic of a data object. This definition can be used to define a linguistic attribute as a property or characteristic of a text token. Linguistic attributes are outputs of preprocessing tasks on text tokens. That means each linguistic attribute can also be seen as a label which is the output of a preprocessing task.

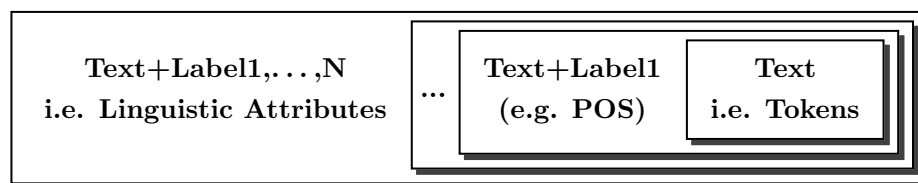


Figure 6.1: Preprocessing

Figure 6.1 illustrates the concept of linguistic features. Tokens of a text can be seen as the data object given to different preprocessing tasks to calculate linguistic attributes, i.e.:

Step 1. $\text{Label1} \leftarrow \text{Task1}(\text{Text})$

$\text{Attribute1} \leftarrow \text{Label1}$

Step 2. $\text{Label2} \leftarrow \text{Task2}(\text{Text}, \text{Attribute1})$

$\text{Attribute2} \leftarrow \text{Label2}$

Step 3. $\text{Label2} \leftarrow \text{Task2}(\text{Text}, \text{Attribute1}, \text{Attribute2})$

...

Step N. $\text{LabelN} \leftarrow \text{TaskN}(\text{Text}, \text{Attribute1}, \dots, \text{AttributeN-1})$

And by applying measurement scales on the linguistic attributes the linguistic features are obtained. In chapter 5 a formalism was presented that can be used to define and use the measurement scales for NLP. It is clear that the more the number of preprocessing tasks, the richer the feature space.

Figure 6.2 compares the concepts of shallow vs. linguistic (rich) feature spaces.

6.3 Case Study: Word Alignment

In this section, I present *2D-Linking*, a new unsupervised method for word alignment based on association scores between words in a bitext (term-frequency-based approach). 2D-Linking can align m-to-n units. It is very efficient because it requires only two passes over the data and less memory than other methods. It

Text				
Token1	Token2	...	TokenN	Shallow
T1-Attribute1	T2-Attribute1	...	TN-Attribute1	Linguistic
T1-Attribute2	T2-Attribute2	...	TN-Attribute2	
...	
T1-AttributeM	T2-AttributeM	...	TN-AttributeM	

Figure 6.2: Shallow vs. linguistic (Rich) features

will be shown that 2D-Linking is superior to competitive linking and as good as or better than symmetrized IBM Model 1 in terms of alignment quality. It also supports trading off precision against recall.

Word alignment, the task of establishing correspondences between words in a bitext (i.e., a sentence-aligned parallel corpus), is an important problem with applications in statistical machine translation, the automatic generation of bilingual dictionaries and cross-language information retrieval. According to Och and Ney [2003], there are two general approaches to computing word alignments: statistical and heuristic methods. In statistical alignment methods [Brown et al., 1993], $Pr(T|S)$ is written in terms of the conditional probability $Pr(T, a|S)$ as:

$$Pr(T|S) = \sum_a Pr(T, a|S) \quad (6.1)$$

Here, the alignment a describes a mapping from the positions of the words in the source text S to the positions in the target text T .

The heuristic methods are considerably simpler. Generally speaking, they try to align each word according to the associative information of source-target word pairs. This information can be provided using different methods. The baseline method will be a simple heuristic model that is called *maximum linking*. For a given target word t_j , maximum linking selects the source word s_i with the highest association score V :

$$s_i = \arg \max_{s'_i \in S} \{V(s'_i, t_j)\} \quad (6.2)$$

A refinement of this linking method is competitive linking which removes each linked word pair from the association score matrix (see Section 6.5.1).

Just as another simple heuristic model does, I also define θ -linking. θ -linking selects all links with association scores greater than a threshold θ :

$$\{s_i\} = \{s'_i \in S | V(s'_i, t_j) > \theta\} \quad (6.3)$$

The threshold θ helps to keep weak candidates out of the search space.

In this thesis I introduce a new method of linking, *Max- θ -Linking*, as well as a new method for calculating an association score matrix, *two-dimensional normalization* or 2DN. I call the combination of Max- θ -Linking and 2DN *2D-Linking*. The advantages of 2D-Linking are as follows:

- It is very efficient. It can be run on standard hardware for arbitrarily large bitexts. And it can be easily distributed on multiple machines.
- It is more accurate than competitive linking and as accurate or more accurate than IBM Model 1, the current method of choice for initializing training of statistical machine translation models.
- It can align m-to-n units, unlike the IBM models, which can only align 1-to-n units.
- It can easily trade off precision vs. recall in word alignment. This is important for many applications of word alignment, in particular for cross-language information retrieval (which in many scenarios requires high precision of word alignments, [Kraaij, 2004]) and machine translation (which usually requires high recall in word alignments, [Fraser and Marcu, 2007a]).

6.3.1 Related Work

Statistical alignment models depend on a set of unknown parameters that must be learned from training data. IBM Model 1 is a particularly simple instance of the framework presented in Equation 6.1. This model assumes a uniform prior probability, where all choices of target words generated by a word in the source sentence are equally probable. The translation probability $\text{tr}(t_j|s_i)$ of the generated target word depends only on the generating source word. Brown et al. [1993] describe the model as follows:

$$Pr(T|S) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l \text{tr}(t_j|s_i) \quad (6.4)$$

Equation 6.4 is the probability estimate of a target sentence T with length m , given a source sentence S with length l . Due to this equation, IBM Model 1 – like many other word-alignment models – chooses for each target word exactly one source word. For more details see [Brown et al., 1993].

IBM Model 1 is mostly used for the initialization of the other IBM models. Structurally, it cannot have m-to-n links. The model can be improved upon. Moore [2004] has introduced an improvement of IBM Model 1 that tries to control the trade-off between precision and recall by adding additional null words to the source sentence. Like Model 1, Moore’s model requires several passes through the data (whereas 2D-Linking only needs two) and cannot model m-to-n links.

There are also some more advanced unsupervised models such as the HMM word alignment model [Vogel et al., 1996], IBM Model 4 [Brown et al., 1993], the Joint (phrase) model [Marcu and Wong, 2002] and Matrix Factorization [Goutte et al., 2004]. But they are much more expensive to calculate. In addition, models like HMM and Model 4 suffer from 1-to-n structure, while the Joint model only allows units consisting of consecutive words. LEAF [Fraser and Marcu, 2007b] directly models m-to-n structure, but is expensive to compute. Alignment by agreement [Liang et al., 2006a] is another approach, in which two asymmetric models are trained jointly to maximize a combination of the likelihood of the data. This approach is more expensive than training the two asymmetric models independently. It is also biased towards high precision alignment structure, as the posterior of both 1-to-n models must be high for a link to be selected.

One of the best known heuristic models is competitive linking [Melamed, 1997]. In competitive linking, the word pair with the highest association score is aligned first (this is similar to maximum linking, Eq. 6.2). Then the corresponding row and column are deleted from the alignment matrix. This process is repeated until either all rows have been deleted or all columns have been deleted. The advantage of this method is its ability to filter out most indirect associations – words that have high association scores but are not translations of each other. The main problem with competitive linking is its inability to produce m-to-n links. It will be shown that 2D-Linking is also able to effectively handle indirect associations. But, importantly, it can find m-to-n links.

2D-Linking is most similar to the approach by Tiedemann [2003]. He defines the word alignment clue as a score which indicates an association between source-

target words by considering various features derived from co-occurrence statistics. The search algorithm though, is more similar to competitive linking, but it handles 1-to-n and m-to-1 alignments as well. 2D-Linking could be extended to use Tiedemann’s scores in a straightforward manner since it can operate on any association measure.

Och and Ney [2003] and Koehn et al. [2003] defined a heuristic procedure that produces an m-to-n alignment. Start the procedure by generating the predicted 1-to-n alignment in the direction source to target. In this alignment one source word aligns to zero or more target words. Call the resulting alignment A1. Generate the predicted m-to-1 alignment in the direction target to source. In this alignment one target word aligns to zero or more source words. Call the resulting alignment A2. Combine A1 and A2 into an m-to-n alignment using a symmetrization heuristic. I consider the following three symmetrization heuristics in this thesis:

(1) The “Union” heuristic takes the union of the links in the A1 and A2 alignments. This results in an alignment having m-to-n discontinuous structure.

(2) The “Intersection” heuristic takes the intersection of the links in the A1 and A2 alignments. This will result in a 1-to-1 alignment structure.

(3) The “Refined” heuristic starts from the “Intersection” 1-to-1 alignment, and adds some of the links present in the “Union” m-to-n discontinuous alignment following the algorithm defined by Och and Ney [2003]. This results in an alignment containing 1-to-n and m-to-1 correspondences, but importantly the words in a correspondence must be consecutive, so this is not as general as the “Union” heuristic.

I will show that 2D-Linking has better alignment quality than symmetrized Model 1.

6.3.2 Associative Information

In the heuristic models, the way association between words is calculated is of central importance. There are many different methods that can be used, for example Dice coefficient scores [Dice, 1945], point-wise mutual information (PMI) [Manning and Schütze, 1999], log-likelihood-ratio (LLR) [Moore, 2004], or expected mutual information (MI) [Cover and Thomas, 1991]. I now discuss the shortcomings of Dice and PMI.

Several approaches to alignment have used the Dice coefficient (e.g., [Zhang et al., 2003]). It is defined as follows [Dice, 1945]:

$$Dice(s, t) = \frac{2C_{ST}(s, t)}{C_S(s) + C_T(t)} \quad (6.5)$$

where C_S and C_T are the word frequencies in the two languages, and C_{ST} is the number of co-occurrences of the two words in sentence pairs of the bitext.

PMI between the source word s and the target word t is calculated as follows:

$$PMI(s, t) = \log_2 \frac{P(s, t)}{P(s) \times P(t)} \quad (6.6)$$

PMI and Dice are similar in that they only consider the number of word occurrences and the number of occurrences of the word pair – but no “non-occurrence” information. Moore [2004] uses the log-likelihood-ratio (LLR):

$$LLR(s, t) = \sum_{i_s \in \{0,1\}} \sum_{i_t \in \{0,1\}} C(s = i_s, t = i_t) \times \log_2 \frac{P(s = i_s | t = i_t)}{P(s = i_s)} \quad (6.7)$$

The main advantage of LLR is that it pays attention to the entire “space” of co-occurrence. That is, Equation 6.7 considers all cases where the corresponding words do or do not occur in the respective target and source sentences. This reduces the association scores of word pairs which are not translations of each other (cf. table 6.1).

Expected mutual information (MI) is a normalized formulation of LLR. MI is calculated as follows: [Cover and Thomas, 1991]

$$MI(s, t) = \sum_{i_s \in \{0,1\}} \sum_{i_t \in \{0,1\}} P(s = i_s, t = i_t) \times \log_2 \frac{P(s = i_s, t = i_t)}{P(s = i_s) \times P(t = i_t)} \quad (6.8)$$

MI is to be preferred over LLR because MI is the standard measure for calculating the mutual dependence of two random variables in information theory.

German word	Translation	MI $\times 10^6$	PMI	Dice $\times 10^3$	Evaluation
diesem	this	34.11	1.54	9.71	+
dieser	this	28.05	1.31	10.06	+
dieses	this	27.70	1.52	8.29	+
diese	this	17.53	1.02	9.43	+
dies	this	13.74	1.25	5.94	+
diesen	this	6.60	1.05	4.00	+
ich	I	6.30	0.40	12.68	-
der	the	4.41	-0.19	13.62	?
das	the	4.23	0.33	12.25	+
ist	is	4.08	0.34	11.62	-
parlament	parliament	3.85	0.64	5.33	-
namen	names	3.35	-1.45	0.61	-
die	the	3.14	-0.15	14.18	?
thema	subject	3.07	1.08	1.89	-
hier	here	3.04	0.67	4.05	-
haus	house	2.83	1.45	1.07	-
heute	today	2.11	0.65	3.11	-
frage	question	1.98	0.62	3.15	-
wir	we	1.95	0.23	11.16	-
hauses	house	1.86	1.57	0.62	-

Table 6.1: Association values of the English word *this* and its corresponding German words. Translations are marked as correct (+), incorrect (-) or context-dependent (?).

In table 6.1, Dice, MI and PMI association scores of a number of German words with the English word *this* are shown, sorted according to MI. The table shows that PMI is correlated with MI. However, its maximum is *hauses*, which is not the correct choice. Also, it incorrectly ranks *haus* higher than the correct translations *dieser*, *diese*, *dies* and *diesen*. Similarly, Dice has *die* as maximum and ranks *ich* higher than the correct translations *diesem*, *dieser*, *dieses*, *diese*, *dies* and *diesen*. I attribute the better performance of MI in this case to the fact that it takes into account non-occurrence counts whereas Dice and PMI do not.

6.3.3 2D-Linking

2D-Linking performs two main steps for alignment. To prevent linking of indirect associations, it first normalizes the association scores by dividing each raw score by the sum of the scores of its row and of its column. This produces two sets of normalized probability distributions, one for the rows and one for the columns of the matrix of association scores. These two normalized sets are averaged to produce the new matrix of association scores. The second step in 2D-Linking links the word pairs according to the Max- θ -Linking method.

I now describe 2D normalization (2DN) in detail. The input to 2DN is a matrix M of association scores. Then a row-normalized matrix R and a column-normalized matrix C are computed by dividing each element by the sum of its row and its column, respectively:

$$R_{ij} = \frac{M_{ij}}{\sum_{j'=1}^m M_{ij'}} \times 100 \quad (6.9)$$

$$C_{ij} = \frac{M_{ij}}{\sum_{i'=1}^l M_{i'j}} \times 100 \quad (6.10)$$

The two matrices are then averaged to build the decision-matrix (D):

$$D_{ij} = \frac{R_{ij} + C_{ij}}{2} \quad (6.11)$$

At the linking step, the algorithm uses D to compute the binary word alignment matrix (A) of a sentence pair according to the Max- θ -Linking method, as follows:

- As alignment $A_{ij} = 1$ for the source word s_i , choose the target word t_j with the largest association score in D that is greater than the linking threshold θ :

$$A \left[i, \arg \max_j \{D_{ij} | D_{ij} > \theta\} \right] = 1 \quad (6.12)$$

- As alignment $A_{ij} = 1$ for the target word t_j , choose the source word s_i with the largest association score in D that is greater than the linking threshold θ :

$$A \left[\arg \max_i \{D_{ij} | D_{ij} > \theta\}, j \right] = 1 \quad (6.13)$$

An alternative linking is θ -Linking, which does not impose the maximum constraint:

- As alignment $A_{ij} = 1$ for the source word s_i , choose all target words t_j with the association score in D greater than the linking threshold θ :

$$A[i, j] = 1 \text{ iff } D_{ij} > \theta \quad (6.14)$$

The basic idea of this new method of calculation of association scores is the normalization of columns and rows of association-score matrix and averaging them to build a matrix D of 2DN association scores. In D , the association scores of the indirect associations are de-emphasized, and the association scores of the true translations are emphasized. The example in table 6.2 shows how this works. In this example, **she** must be linked to **sie**, and **has** to **hat**. By comparing association scores, **she** is linked incorrectly to **hat** (i.e., indirect association link). But after 2D normalization, the linking score of **she** and **hat** is reduced and **she** is correctly linked to **sie**. **Example:** To illustrate 2D-Linking, consider the German sentence *Ich beziehe mich auf Punkt 11 des Arbeitsplans* ('I refer to item 11 on the order of business.'). Table 6.3 shows the MI matrix M , and table 6.4 shows the decision matrix D derived from M and the alignment computed by 2D-Linking. Table 6.5 shows the same sentence pair aligned with competitive linking. An example that shows the advantage of 2D-Linking is the word **Arbeitsplan**, which is linked to the words **order** and **business**.

Even though the alignment computed by 2D-Linking is better, it is not error-free. The alignment **auf** \leftrightarrow **on** is an error. In addition, **of** should probably be

	sie	hat
she	21	215
has	2	6916

	sie	hat
she	50%	47%
has	4%	98%

Table 6.2: Example for indirect associations. Scores are $MI \times 10^6$. Left: raw associations, right: associations after 2D normalization.

	Ich	beziehe	mich	auf	Punkt	11	des	Arbeitsplan	.
I	[211.84]	0.73	[48.24]	0.13	2.01	0.01	4.53	0.00	4.78
refer	0.91	[1.81]	0.88	2.38	0.04	0.12	0.00	0.13	0.03
to	5.06	0.06	[1.51]	[3.46]	0.20	0.13	4.00	0.06	0.04
item	0.41	0.09	0.25	0.18	[3.03]	0.02	5.84	0.26	0.76
11	0.00	0.17	0.00	0.01	0.14	[20.92]	0.02	0.15	0.04
on	0.05	0.16	0.29	[10.91]	0.59	0.12	14.92	0.04	2.46
the	1.14	0.00	0.83	0.52	0.09	0.00	[38.72]	0.01	4.88
order	0.06	0.01	0.00	0.00	0.02	0.00	0.02	[0.16]	0.09
of	4.86	0.00	1.11	0.41	0.18	0.02	[34.15]	0.11	2.49
business	0.02	0.11	0.05	0.15	0.00	0.04	0.01	[0.70]	0.00
.	8.08	0.01	0.69	0.00	0.62	0.01	3.81	0.01	[92.59]

Table 6.3: The association-score matrix M for the translation example. Scores are expected $MI \times 10^6$. Scores that generate links in 6.4 (after normalization) or in 6.5 (raw scores) are in bold. Word-by-word translation: **Ich** \rightarrow I; **beziehe** \rightarrow refer, relate; **mich** \rightarrow me, myself; **auf** \rightarrow on, in, at, by; **Punkt** \rightarrow point, dot, spot; **des** \rightarrow of; **Arbeitsplans** \rightarrow work schedule.

linked to *Arbeitsplan*. Some errors of this type could be fixed by using word position or phrase alignment. This will be briefly discussed in section 6.6.

6.3.4 Evaluation

For the calculation of association scores, the Europarl English-German parallel corpus was used. It consists of about 18.7 million German words, 17.8 million English words and 650,000 sentence pairs.

The linking threshold ($0 \leq \theta \leq 100$) can be selected to trade off precision and recall. The higher the θ , the higher the precision and the lower the recall.

	Ich	beziehe	mich	auf	Punkt	11	des	Arbeitsplan	.
I	[84.5]	11.7	[53.6]	0.4	14.9	0.0	3.0	0.0	3.1
refer	7.4	[43.1]	7.8	25.4	0.7	1.2	0.0	5.0	0.2
to	18.5	1.2	6.6	[21.5]	2.1	0.7	15.6	2.1	0.1
item	2.0	1.8	1.4	1.3	[35.8]	0.1	29.7	9.3	3.9
11	0.0	3.0	0.0	0.1	1.4	[97.7]	0.1	4.8	0.1
on	0.1	2.8	0.8	[48.5]	5.2	0.5	32.3	1.3	5.3
the	1.5	0.0	1.7	2.0	0.8	0.0	[60.2]	0.4	7.5
order	7.7	1.7	0.5	0.0	3.3	0.0	2.6	[26.7]	12.7
of	6.7	0.0	2.3	1.6	1.5	0.1	[55.5]	3.5	4.0
business	0.9	7.1	2.3	7.3	0.2	1.9	0.4	[53.5]	0.1
.	5.6	0.2	1.0	0.0	4.8	0.0	3.6	0.3	[86.6]

Table 6.4: The decision matrix D for the translation example. The table shows the association scores normalized by 2D-Linking. Scores that are selected as maxima in Equations 6.12 and 6.13 and give rise to alignments are in bold.

	Ich	beziehe	mich	auf	Punkt	11	des	Arbeitsplan	.
I	[211.84]	0.73	48.24	0.13	2.01	0.01	4.53	0.00	4.78
refer	0.91	[1.81]	0.88	2.38	0.04	0.12	0.00	0.13	0.03
to	5.06	0.06	[1.51]	3.46	0.20	0.13	4.00	0.06	0.04
item	0.41	0.09	0.25	0.18	[3.03]	0.02	5.84	0.26	0.76
11	0.00	0.17	0.00	0.01	0.14	[20.92]	0.02	0.15	0.04
on	0.05	0.16	0.29	[10.91]	0.59	0.12	14.92	0.04	2.46
the	1.14	0.00	0.83	0.52	0.09	0.00	[38.72]	0.01	4.88
order	0.06	0.01	0.00	0.00	0.02	0.00	0.02	0.16	0.09
of	4.86	0.00	1.11	0.41	0.18	0.02	34.15	0.11	2.49
business	0.02	0.11	0.05	0.15	0.00	0.04	0.01	[0.70]	0.00
.	8.08	0.01	0.69	0.00	0.62	0.01	3.81	0.01	[92.59]

Table 6.5: Alignment computed by competitive linking for the translation example. Bold scores indicate an alignment of the two corresponding words.

For the evaluation of 2D-Linking, I use a manually aligned parallel corpus that was provided by Callison-Burch [2007]. This gold standard is an annotation of a part of the Europarl English-German parallel corpus. I used 120 sentence pairs which consisted of 3,564 English words, 3,320 German words and 3,223 gold standard links.

I compare the output of the alignment methods with the gold standard using different trade-offs between precision and recall in the F-measure formula as follows:

$$F_\alpha = 1 / \left(\frac{\alpha}{precision} + \frac{1 - \alpha}{recall} \right) \quad (6.15)$$

Table 6.6 and 6.7 show the results of the experimental evaluation. I first compare our systems with competitive linking. The best association statistic for competitive linking is PMI (table 6.6, line 3). Compared with competitive linking, 2D-Linking (Max- θ -Linking + 2DN) results (table 6.6, lines 25–24) have better F. The largest gap is when F-Measures with small α are considered (i.e. where F-Measure is biased towards recall). The reason for this is that 2D-Linking can create m-to-n alignments, while competitive linking is restricted to 1-to-1 alignments.

Next I compare competitive linking (table 6.6, lines 15–24) with symmetrized IBM Model 1 (table 6.6, lines 4–8). At higher values of α , 2D-linking has the same performance as the “refined” Model 1 symmetrization. However, as α decreases, the story changes and 2D-Linking has superior performance. At $\alpha = 0.4$, “refined” is the best IBM Model 1 symmetrization, but it is outperformed by 2D-Linking. At $\alpha = 0.2$, “union” is the best IBM Model 1 symmetrization, but it is also outperformed by 2D-Linking.

I performed additional experiments to try to understand the contributions of the different components of thresholded 2D normalization. The first set of additional experiments I compared θ -Linking (table 6.6, lines 9–14) and Max- θ -Linking (table 6.6, lines 15–24). Both linking techniques can create m-to-n alignments. For small θ s, θ -Linking has higher recall than Max- θ -Linking but at a cost: precision is too high. For large θ s, θ -Linking and Max- θ -Linking have similar results, as the filtering by the “maximum rule” (compare Equations 6.12 and 6.13 with Equation 6.14) no longer has much effect. This shows that

	Alignment Method	F-Measure					
		$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
1	Competitive Linking, DICE	0.30	0.33	0.38	0.43	0.51	0.61
2	Competitive Linking, MI	0.33	0.37	0.41	0.47	0.54	0.64
3	Competitive Linking, PMI	0.40	0.43	0.47	0.52	0.58	0.66
4	IBM Model 1 (Source English)	0.46	0.48	0.50	0.53	0.56	0.59
5	IBM Model 1 (Source German)	0.51	0.52	0.54	0.56	0.57	0.60
6	IBM Model 1 (Union)	0.72	0.52	0.41	0.34	0.29	0.25
7	IBM Model 1 (Intersection)	0.37	0.42	0.48	0.56	0.67	0.84
8	IBM Model 1 (Refined)	0.47	0.51	0.55	0.61	0.67	0.75
9	θ -Linking ($\theta = 0\%$), 2DN(MI)	0.80	0.41	0.28	0.21	0.17	0.14
10	θ -Linking ($\theta = 10\%$), 2DN(MI)	0.65	0.61	0.57	0.54	0.51	0.48
11	θ -Linking ($\theta = 20\%$), 2DN(MI)	0.55	0.56	0.58	0.59	0.60	0.62
12	θ -Linking ($\theta = 30\%$), 2DN(MI)	0.46	0.49	0.54	0.58	0.64	0.71
13	θ -Linking ($\theta = 40\%$), 2DN(MI)	0.42	0.46	0.52	0.58	0.67	0.78
14	θ -Linking ($\theta = 50\%$), 2DN(MI)	0.34	0.39	0.45	0.53	0.64	0.83
15	Max- θ -Linking ($\theta = 0\%$), 2DN(MI)	0.65	0.62	0.59	0.56	0.53	0.51
16	Max- θ -Linking ($\theta = 10\%$), 2DN(MI)	0.62	0.61	0.60	0.59	0.58	0.57
17	Max- θ -Linking ($\theta = 20\%$), 2DN(MI)	0.55	0.57	0.59	0.61	0.63	0.65
18	Max- θ -Linking ($\theta = 30\%$), 2DN(MI)	0.47	0.51	0.55	0.59	0.65	0.72
19	Max- θ -Linking ($\theta = 40\%$), 2DN(MI)	0.42	0.46	0.52	0.58	0.67	0.78
20	Max- θ -Linking ($\theta = 50\%$), 2DN(MI)	0.34	0.39	0.45	0.53	0.64	0.83
21	Max- θ -Linking ($\theta = 60\%$), 2DN(MI)	0.29	0.33	0.40	0.48	0.62	0.87
22	Max- θ -Linking ($\theta = 70\%$), 2DN(MI)	0.22	0.26	0.31	0.40	0.55	0.89
23	Max- θ -Linking ($\theta = 80\%$), 2DN(MI)	0.16	0.19	0.24	0.32	0.47	0.90
24	Max- θ -Linking ($\theta = 90\%$), 2DN(MI)	0.07	0.09	0.11	0.16	0.27	0.91

Table 6.6: F measures for the different combinations of heuristic methods and IBM Model 1. The largest F in each column is in bold. $\alpha = 0$ is recall, and $\alpha = 1$ is precision.

	Alignment Method	Precision	Recall	F Measure ($\alpha = 0.5$)
1	Max- θ -Linking ($\theta = 20\%$), 2DN(MI)	0.65	0.55	0.60
2	Max- θ -Linking ($\theta = 20\%$), 2DN(PMI)	0.59	0.43	0.50
3	Max- θ -Linking ($\theta = 20\%$), 2DN(DICE)	0.69	0.31	0.43
4	Competitive Linking, 2DN(PMI)	0.56	0.33	0.42
5	Competitive Linking, 2DN(MI)	0.64	0.33	0.44

Table 6.7: Precision, recall and F measures for the different combinations of heuristic methods. The largest F is in bold.

the maximum rule is effective for generating both high recall and high precision alignments.

I then examined the effect of the association score (2DN) used on 2D-Linking. I will discuss the results for Max- θ -Linking with $\theta = 20\%$, which has a good trade-off between recall and precision. When PMI was tried, I obtained both lower precision and recall than with MI (table 6.7, lines 1–2). Dice had slightly higher precision and much lower recall than MI (table 6.7, line 3). Max- θ -Linking works best with 2DN(MI).

Finally, I also tried competitive linking with 2DN. This did not help performance (table 6.7, lines 4–5). I believe that the smoothing effect of 2DN is not important when used with competitive linking, given that the competitive linking algorithm deletes the row and column of each link selected, which at each step is always the maximum cell value of the matrix M .

6.3.5 Summary

I have presented 2D-Linking, a new unsupervised method of word alignment. In comparison with competitive linking and IBM Model 1, 2D-Linking gives better results. The linking threshold makes it possible to easily trade off precision against recall. Such a trade-off is important for many applications. For example, cross-language information retrieval requires high precision.

2D-Linking can be based on any definition of association strength, but I have shown that expected mutual information gives better results than using Dice and PMI, two measures that were previously used to compute word alignments.

In 2D-Linking, vocabulary can be partitioned and association scores for each

partition can be computed separately. This makes it possible to distribute the process over different machines – or to run the computations sequentially if memory is scarce.

2D-Linking is easy to implement and it is sufficiently flexible and modular to be combined with other linguistic or statistical methods.

The m-to-n alignment produced by 2D-Linking can be used directly in many applications of word alignment. However, for bootstrapping a statistical machine translation model (such as the IBM models), a probability distribution is needed, rather than a prediction of the best word alignment. This alignment probability distribution can be calculated by normalization of the decision matrix D . In other words, the association scores of a target word are normalized over all its possible translation source words and vice versa to calculate a probability distribution over all lexical connections. This probability distribution can then be used to bootstrap translation models such as the IBM models or a discriminative word alignment model such as [Moore et al., 2006].

6.4 Unsupervised CoRe with A-INF

In this section, A-INF will be presented, which is a new unsupervised method for coreference resolution based on lexical association scores between heads of markables (term-frequency-based approach).

Until recently, most approaches tried to solve the problem of coreference resolution by binary classification, where the probability of a pair of markables being coreferent is estimated from labeled data. Alternatively, a model that determines whether a markable is coreferent with a preceding cluster can be used. For both pair-based and cluster-based models, a well-established feature model plays an important role. Typical systems use a *rich feature space* based on lexical, syntactic and semantic knowledge. The most commonly used features have been described by Soon et al. [2001].

Most existing systems are supervised systems, trained on human-labeled benchmark data sets for English. These systems use linguistic features based on number, gender, person, etc. It is a challenge to adapt these systems to new domains, genres and languages, because significant human labeling effort is usually necessary to get good performance.

To address this challenge, I decided on an *unsupervised approach* (A-INF). It is based on association information that is mined from an unlabeled corpus in a completely unsupervised fashion. The key novelty of my approach is that it considers the actual identity of the words in question.

This approach is reminiscent of “lexical feature”. The term *lexical feature* in coreference resolution literature generally refers to (sub)string and edit distance features. In this thesis, a distinction will be drawn between such string-based lexical features and *lexical information* that makes use of the actual identity of the words in question; e.g., “Daschle” – “senator” is a likely coreference link, but this cannot be determined by only looking at the two strings. String-based lexical features are part of the representation of individual links in standard supervised CoRe. In our approach, word association information is used for *clustering markables* in unsupervised learning. Association information is calculated as *association scores between heads of markables*, as will be described below. I view association information as an example of a *shallow feature space* which contrasts with the rich feature space that is generally used in CoRe.

While this method is shallow, it provides valuable information for CoRe. Consider the pair of markables (*Obama, President*). It is a likely coreference pair, but this information is not accessible to standard CoRe systems because they only use string-based features (often called lexical features), named entity features and semantic word class features (e.g., from WordNet) that do not distinguish, say, *Obama* from *Hawking*.

6.4.1 System Architecture

The top level architecture of our unsupervised model A-INF is identical to our supervised model SUCRE, which was illustrated in Figure 4.1.

In *preprocessing*, markables, tokens and their attributes are extracted from the input text. The key difference between the unsupervised and supervised approaches is in how *pair estimation* is accomplished. Similar to the supervised model SUCRE, pair estimation in A-INF has two forms of learning and classification, which will be presented below.

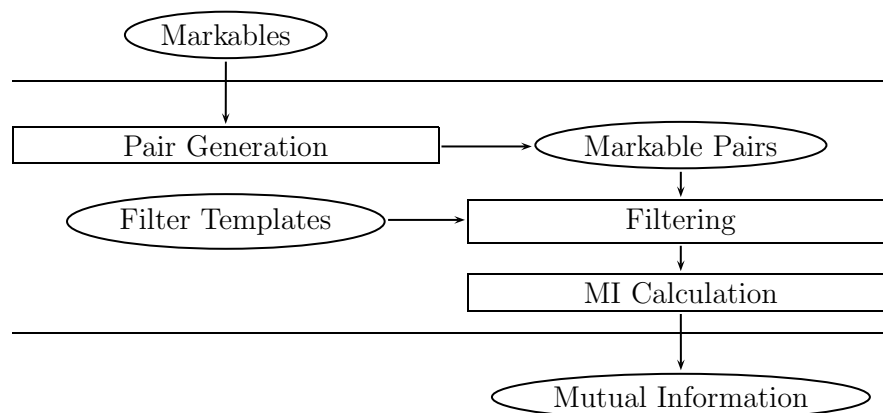


Figure 6.3: Pair Estimation: Learning (i.e. calculation of mutual information).

6.4.2 Learning

Figure 6.3 shows how learning is performed in the pair estimation part of A-INF. In this module there are three sub-modules:

Pair Generation

Due to the fact that A-INF is an unsupervised system, the pair generation for training differs from what we have had for SUCRE.

As mentioned before, McEnery et al. [1997] showed that in 98.68% of cases the antecedent is within a 10-sentence window; hence, in the pair generation step, all possible pairs inside 10 sentences are generated.

Filtering

The feature definition language (see Chapter 5) was used to define the templates according to which the filters are to be calculated. These templates are hard constraints that filter out all cases that are clearly disreferent, e.g., (*he - she*), (*he - they*) or (*I - it*). I used the following filters:

Filter 1: the antecedent of a reflexive pronoun must be in the same sentence.

Filter 2: the antecedent of a pronoun must occur at a distance of at most 3 sentences.

Filter 3: a coreferent pair of a noun and a pronoun or of two pronouns must not disagree in number.

Filter 4: a coreferent pair of two pronouns must not disagree in gender.

In addition to the above described filters, the following filtering rule (it concerns string matching) was also used:

Filter 5: if the head of markable M_2 matches the head of the preceding markable M_1 , then all other pairs for M_2 are ignored in the calculation of association scores.

This additional filter (i.e. 5) is necessary, because an approach without some kind of string matching constraint yields poor results, given the importance of string matching for CoRe. Without filter 5, the performance of A-INF is poor. As will be shown below, even the simple filters 1–5 are sufficient to learn high-quality association scores; this means that the complex features of “deterministic” systems are not needed. However, if such complex features are available, then they can be used to improve performance in our self-trained setting.

MI-Calculation

To calculate the degree of association between two markables, lexical information is learned from a large unlabeled corpus (to be described in section 6.4.6) by computing mutual information (MI) scores between heads of markables.

MI is calculated as follows: [Cover and Thomas, 1991]

$$MI(a, b) = \sum_{i_a \in \{\bar{a}, a\}} \sum_{i_b \in \{\bar{b}, b\}} P(i_a, i_b) \times \log_2 \frac{P(i_a, i_b)}{P(i_a) \times P(i_b)}$$

A key virtue of this approach is that in the classification of links as coreferent/disreferent, the coreference probability p estimated in supervised learning plays exactly the same role as the lexical association score N_{MI} (defined below) that was learned in an unsupervised fashion. For p , it is important that we only consider links with $p \geq 0.5$ as potentially coreferent (see figure 4.14). To be able to impose the same constraint on N_{MI} , a measure is needed that (i) ranges between 0 and 1; and (ii) gives values $N_{MI} \geq 0.5$ to possible links and values $N_{MI} < 0.5$ to unlikely links.

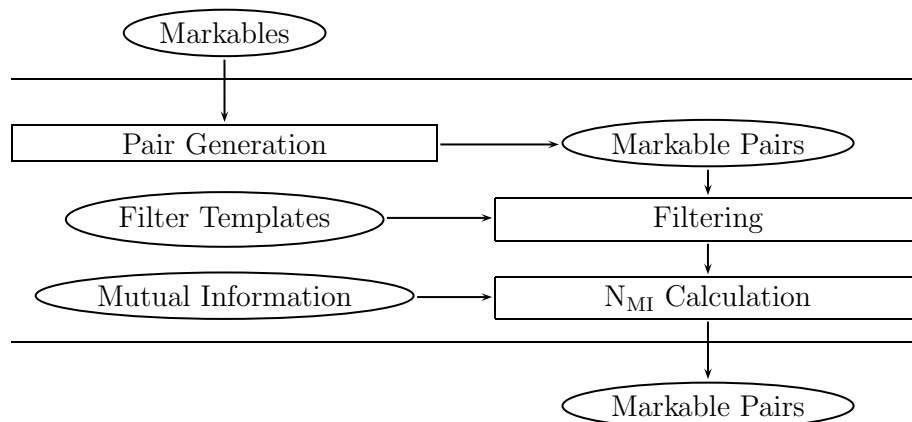


Figure 6.4: Pair Estimation: Classification (i.e. calculation of N_{MI} of markable pairs).

6.4.3 Classification

Figure 6.4 shows the architecture of classification module in the pair estimation part of A-INF. It contains three functional parts: pair generation, filtering and N_{MI} -Calculation.

Pair generation for classification is identical to learning, i.e. all possible pairs inside 10 sentences are generated.

For filtering, filters 1–4 were used which had been introduced before for learning.

N_{MI} -Calculation

Inspired by the 2D-Linking method that was presented in Section 6.3.3, a N_{MI} score was proposed, for which the MI scores were normalized by looking at the maximum values of the two words and taking the average:

$$N(a, b) = \left(\frac{MI(a, b)}{\arg \max_x MI(a, x)} + \frac{MI(a, b)}{\arg \max_x MI(x, b)} \right) / 2 \quad (6.16)$$

In the above equation, the value of N_{MI} indicates how strongly two words are associated. N_{MI} is normalized to ensure $0 \leq N \leq 1$. If a or b has not occurred before, then N_{MI} is set to 0.

Compared to 2D-Linking, where a row-normalized value (matrix R) and a column-normalized value (matrix C) were computed, in N_{MI} -Calculation we compute (i) the normalized value of $MI(a, b)$ over all markable pairs, in which a has participated and has been seen in the training corpus (similar to row-normalization in 2D-Linking); and (ii) the normalized value of $MI(a, b)$ over all markable pairs in which b has participated and has been seen in the training corpus, where $b \in \{ \text{all possible markable pairs within 10 sentence window from } a \}$ (similar to row-normalization in 2D-Linking).

N_{MI} is used for clustering (see figure 6.5) in exactly the same way that p is used: the antecedent is searched for with a maximum association score N_{MI} greater than 0.5 from right to left, starting from the end of the document.

As will be seen below, using N_{MI} -scores acquired from an unlabeled corpus as the only source of information for CoRe performs surprising well.

However, the weakness of this approach is its failure to cover pairs that do not occur in the unlabeled corpus (*lower recall*). A second weakness is that the set of generated links is not restricted to plausible candidates for coreference (*lower precision*). Specifically, the computation of N_{MI} -scores in most cases uses several links for a markable M , including links that could easily be identified as bad because a much better link is available. I therefore pursue a self-training approach that trains a model on a corpus labeled by A-INF. I will discuss the advantages of the self-training approach in section 6.5 by presenting two examples and reporting the results.

6.4.4 Chain Estimation

The main task in *chain estimation* is clustering. Figure 6.5 presents our clustering method, which is used for unsupervised coreference resolution in A-INF. The best predicted antecedent (with coreference probability $p \geq 0.5$) was sought out from right to left; starting from the end of the document. As was mentioned before, due to the fact (investigated by McEnery et al. [1997]) that in 98.68% of cases the antecedent is within a 10-sentence window, I used a window of 10 sentences for search. I have found that limiting the search to a window increases both efficiency and effectiveness.

Chain_Estimation (M_1, M_2, \dots, M_n)

1. Let t be the clustering threshold initialized with 1.
2. Let each markable M_i be in its own cluster $C_i : C_i = \{M_i\}$.
3. Proceed through the markables from the end of the document. For each markable M_j , consider each preceding markable M_i within a window of 10 sentences:
If *Pair_Estimation* (M_i, M_j) $\geq t$ then $C_i = C_i \cup C_j$.
4. Set $t = t - 0.01$
5. If $t \geq 0.5$ go to step 3.

Figure 6.5: Chain Estimation (Clustering) Algorithm in A-INF

6.4.5 Evaluation

I report recall, precision, and F_1 -measure for the following metrics: MUC [Vilain et al., 1995], B³ [Bagga and Baldwin, 1998], CEAF [Luo, 2005].

For all experiments I performed two-tailed significance tests (t-test, $df = 4$) with the null hypothesis that there is no significant difference in the evaluation scores between the randomly selected disjoint subsets of the test set. A p-value of .01 was used throughout.

6.4.6 Data Sets

For computing word association, a corpus of about 63,000 documents from the 2009 English *Wikipedia* (the articles that were larger than 200 bytes) was used. This corpus consists of more than 33.8 million tokens; the average document length is 500 tokens. The corpus was parsed using the Berkeley parser [Petrov and Klein, 2007]. All sentences with no parse output were ignored. The number of detected markables (all noun phrases extracted from parse trees) is about 9 million.

[Stoyanov et al., 2010] argued that the reported CoRe scores vary wildly across data sets, evaluation metrics, and system configurations. Due to the difficulties they outlined, our result can be directly compared only to the available result of other systems on similar data sets and condition. The unsupervised model A-INF is evaluated using *ACE (Phase 2)*¹ benchmark because this data set is the most widely used CoRe benchmark and is used by the systems that are most comparable to our approach. The corpus is composed of three data sets from three different news sources. Below is the number of test documents for each:

1. Broadcast News (BNEWS) with 51 documents.
2. Newspaper (NPAPER) with 17 documents.
3. Newswire (NWIRE) with 29 documents.

I report results for *true markables* (markables extracted from the answer keys) in order to be able to compare with other systems that use true markables.

Table 6.8 compares the unsupervised model A-INF to P&D [Poon and Domingos, 2008] on ACE-2 and to Ng [Ng, 2008] and H&K² [Haghighi and Klein, 2007] on ACE2003. These are the best and most recent evaluation results on the ACE-2 and ACE-2003 data sets for unsupervised learning.

As can be seen from the table, A-INF F_1 scores are always below those of P&D. Comparing to Ng and H&K, A-INF F_1 score differences range from +5.9 (MUC) to -6.4 (CEAF).

A clear benefit of the model A-INF could not be identified from these results, but as will be shown later, if A-INF is used for bootstrapping of a supervised system, performance is significantly improved.

Learning Curve

Figure 6.6 presents MUC scores of A-INF as a function of the number of Wikipedia articles used in unsupervised learning. It is clear that a small number of input articles (e.g., 100) results in low recall and high precision. When the number of

¹I used two variants of ACE (Phase 2): ACE-2 and ACE2003.

²I report numbers for the better performing Pronoun-only Saliency variant of H&K proposed by Ng [2008].

CHAPTER 6. UNSUPERVISED COREFERENCE RESOLUTION

	MUC			B ³			CEAF		
	ACE-2								
BNEWS	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	68.3	66.6	67.4	70.3	65.3	67.7	–	–	–
A-INF	60.8	61.4	61.1	55.5	69.0	61.5	52.6	52.0	52.3
NWIRE	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	67.7	67.3	67.4	74.7	68.8	71.6	–	–	–
A-INF	62.4	57.4	59.8	59.2	62.4	60.7	46.8	52.5	49.5
NPAPER	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	69.2	71.7	70.4	70.0	66.5	68.2	–	–	–
A-INF	60.6	56.0	58.2	52.4	60.3	56.0	38.9	44.0	41.3
	ACE2003								
BNEWS	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
H&K	68.3	56.8	62.0	–	–	–	59.9	53.9	56.7
Ng	71.4	56.1	62.8	–	–	–	60.5	53.3	56.7
A-INF	60.9	64.9	62.8	50.9	72.5	59.8	53.8	49.4	51.5
NWIRE	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
H&K	66.2	46.8	54.8	–	–	–	62.8	49.6	55.4
Ng	68.3	47.0	55.7	–	–	–	60.7	49.2	54.4
A-INF	62.7	60.5	61.6	54.8	66.1	59.9	47.7	50.2	49.0

Table 6.8: Scores for A-INF on ACE-2 and ACE2003 and scores for comparable coreference systems.

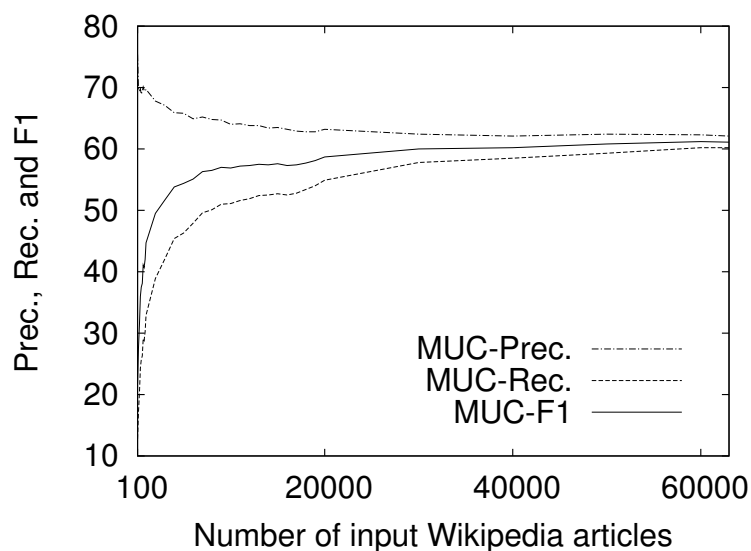


Figure 6.6: MUC learning curve for A-INF.

input articles is increased, recall rapidly increases and precision rapidly decreases up to about 10,000 articles. Increase and decrease continue more slowly after that. F_1 increases throughout because lower precision is compensated by higher recall. This learning curve demonstrates the importance of the size of the corpus for A-INF.

6.5 Unsupervised CoRe with UNSEL

In this section, an unsupervised framework will be presented that bootstraps a complete coreference resolution (CoRe) system from *word associations* mined from a large unlabeled corpus.

As was shown before, word associations are useful for CoRe – e.g., the strong association between *Obama* and *President* is an indicator of likely coreference. Association information has so far not been used in CoRe, because it is sparse and difficult to learn from small labeled corpora. Using a self-training framework, I trained a decision tree on a corpus that is automatically labeled using word associations, i.e. bootstrapping by A-INF. This unsupervised self-trained system has better CoRe performance than other learning approaches that do not use manually labeled data.

Self-training approaches usually include the use of some manually-labeled data. This was not the case in our study. Our self-trained system is not trained on any manually-labeled data and is therefore a completely unsupervised system. Although training on automatically labeled data can be viewed as a form of supervision, we in this thesis the term *supervised system* is reserved for systems that are trained on manually-labeled data.

6.5.1 Related Work

A considerable amount of literature has been published on CoRe. There are three main approaches to this task: supervised, semi-supervised (or weakly supervised) and unsupervised. The term *semi-supervised* will be used for approaches that use some amount of human-labeled data.

Müller et al. [2002] used co-training for coreference resolution, which is a semi-supervised method. Co-training puts features into disjoint subsets when learning from labeled and unlabeled data and tries to leverage this split for better performance. Ng and Cardie [2003] conducted a series of trials in which they mixed a naive Bayes classifier as bootstrapping algorithm with a decision list (DL) algorithm in a multiple-learner framework. They reported that their self-training algorithm performed slightly better than co-training. They argued that, for the CoRe task, the multiple learner approach is a better choice than the multiple view approach of co-training. Our self-trained model combines multiple learners (A-INF and SUCRE) and multiple views (shallow/rich information). A key difference from Müller et al. [2002] and Ng and Cardie [2003] is that I did not use any human-labeled data.

Turning to unsupervised approaches, Haghighi and Klein [2007] proposed a generative Bayesian model with a performance only slightly below that of supervised systems. Poon and Domingos [2008] provided in-depth analysis of this work, showing drawbacks regarding sampling approach, extensibility, cluster size and cluster number that grow logarithmically with the number of data points. They introduced an unsupervised system in the framework of Markov logic. Ng [2008] presented a generative model that views coreference as an EM clustering process.

Recent work by Haghighi and Klein [2009] and Raghunathan et al. [2010]

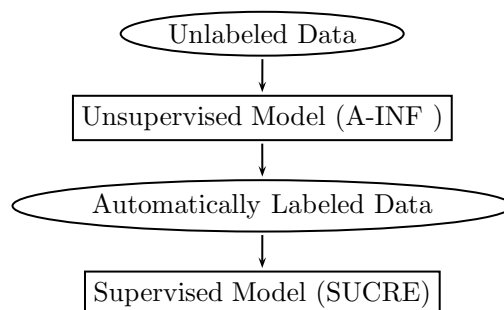


Figure 6.7: System Architecture of UNSEL (Unsupervised Self-Trained Model)

challenges the appropriateness of machine learning methods for CoRe. These researchers show that a *deterministic* system (essentially a rule-based system) that uses a rich feature space including lexical, syntactic and semantic features can substantially improve CoRe performance. However, a limitation of this approach is that it requires substantial effort to encode the large number of deterministic constraints that guarantee good performance. Additionally, it is unlikely to perform well with simpler features. Moreover, these systems are not adaptable (since they are not machine-learned) and may have to be completely rewritten for each new domain, genre and language. Consequently, the performance will not be compared with these deterministic systems. It is unsurprising that a deterministic system carefully constructed for a narrow use case performs better than an automatically learned system. Both “deterministic” and machine learning approaches are promising, but for different application scenarios.

6.5.2 System Architecture

Figure 6.7 illustrates the system architecture of our unsupervised self-trained coreference resolution system (UNSEL). Oval nodes are data, and boxed nodes are processes. I take a self-training approach to coreference resolution: first the corpus is labelled using the unsupervised model A-INF and then the supervised model (SUCRE) is trained on this automatically labeled training corpus. The procedural form of our self-training approach is presented in Figure 6.8.

Even though I trained on a labeled corpus, the labeling of the corpus is produced in a completely automatic fashion, without recourse to human labeling.

```

procedure SelfTrainigUNSEL( $W, U$ )
  1.  $W$  is a very large unlabeled corpus
  2.  $L$  is labeled data,  $U$  is unlabeled data
  3.  $\text{classifier}_{A-INF} \leftarrow \text{train}_{A-INF}(W)$ 
  4.  $L \leftarrow \text{label}(U, \text{classifier}_{A-INF})$ 
  5.  $\text{classifier}_{UNSEL} \leftarrow \text{train}_{SUCRE}(L)$ 
  6. return  $\text{classifier}_{UNSEL}$ 

```

Figure 6.8: Self-Training in UNSEL (Unsupervised Self-Trained Model).

Thus, it is an unsupervised approach. I call this approach self-training here, even though self-training most often refers to an approach that uses both human and automatic labeling.

Our architecture is very flexible; in particular, as will be explained presently, it can be easily adapted for supervised as well as unsupervised settings. A key characteristic of our architecture is that it is modular: different algorithms can be employed for the sub-tasks of CoRe.

To summarize, in order to generate links for training, the system takes a positive training instance for a coreferent markable pair (M_i, M_j) if there is no intervening coreferent markable, and negative training instances for the markable M_i and all markables M_k disreferent with M_i that occur before M_j (i.e. $i < k < j$). For testing, it generates all possible links inside a window of 10 sentences. The filtering step, filters (i)–(iv), was discussed above. Then a feature vector was calculated for each generated link that survived filtering. Our features are similar to those described by Soon et al. [2001]: string-based features, distance features, span features, part-of-speech features, grammatical features, and agreement features. The total number of features is 20.

As was described in chapter 4, the classification method I use is a decision tree classifier [Quinlan, 1993].³ The decision tree is trained on the training set to

³I also tried support vector machines and maximum entropy models, but they did not

estimate the coreference probability p for a link and then applied to the test set. Note that, as is standard in CoRe, filtering and feature calculation are exactly the same for training and test, but link generation is different, as was described above.

6.5.3 Evaluation

As for the A-INF system, I report recall, precision, and F_1 -measure for the following metrics: MUC [Vilain et al., 1995], B³ [Bagga and Baldwin, 1998], CEAF [Luo, 2005]. For all experiments two-tailed significance tests (t-test, $df = 4$) were performed, with the null hypothesis that there is no significant difference in the evaluation scores between the randomly selected disjoint subsets of the test set. A p-value of .01 was used throughout.

6.5.4 Data Sets

Unsupervised, supervised and self-trained models were evaluated on *ACE (Phase 2)* benchmark, because this data set is the most widely used coreference resolution benchmark and is used by the systems that are most comparable to our approach (see section 6.4.6).

In addition, I used the recently published *OntoNotes* coreference benchmark for evaluation. *OntoNotes* is an excerpt of news from the *OntoNotes Corpus Release 2.0*. The advantage of *OntoNotes* is that it contains two parallel annotations: (i) gold standard manual annotations of the preprocessing information, which I call *gold setting*, and (ii) automatically predicted annotations of the preprocessing information, which I call *automatic setting*. The automatic setting reflects the situation a CoRe system faces in reality; in contrast, the gold setting can be considered less realistic.

Another important consideration in CoRe evaluation is the issue of markable detection. In a real application, there is no access to true markables, so an evaluation on system markables (markables automatically detected by the system) is required. However, reporting CoRe numbers on system markables alone is not sufficient either, since CoRe evaluation measures do not necessarily punish a

perform better.

system for not or partly recognizing markables – one can potentially game the system by recognizing only markables that are easy to cluster into coreference chains and call them system markables. Thus, a system should be able to measure the quality of its markable detection sub-task according to a gold standard (true markables). OntoNotes was used in this thesis to perform what I view as a complete and realistic evaluation that takes markable detection into account. I use two suggested configurations provided in SemEval-2010 [Recasens et al., 2010]: (i) the gold setting with the true markables, and (ii) the automatic setting with the system markables.

In the experiments done for the A-INF model, the Wikipedia corpus was used to compute lexical information, and then the model was evaluated on the test part of ACE and OntoNotes. For the experiments related to the UNSEL model, its unsupervised subsystem A-INF (which uses lexical information calculated from Wikipedia) was used to automatically label the training part of ACE and OntoNotes. Then for each data set, the supervised subsystem of UNSEL (i.e., SUCRE) was trained on its automatically labeled training part and then evaluated on its test part. Finally, for SUCRE experiments, human labeled training parts were used and evaluated using the corresponding human labeled test parts.

6.5.5 Results

Table 6.9 compares our unsupervised self-trained model UNSEL and unsupervised model A-INF to P&D [Poon and Domingos, 2008] on ACE-2 and to Ng [Ng, 2008] and H&K⁴ [Haghighi and Klein, 2007] on ACE2003. These are the best and most recent evaluation results on the ACE-2 and ACE-2003 data sets for unsupervised learning. In addition, it presents the results of the supervised system SUCRE for each data set. The SUCRE results will be discussed later in this section.

As can be seen from the table, A-INF F_1 scores are always below those of P&D, but precision and F_1 scores of UNSEL are always better. Comparing to Ng and H&K, A-INF F_1 scores range from +5.9 (MUC) to -6.4 (CEAF), but UNSEL recall and F_1 scores are always better. These results suggest that the self-training part significantly improves the performance. To explain this, I present

⁴I report numbers for the better performing Pronoun-only Saliency variant of H&K proposed by [Ng, 2008].

CHAPTER 6. UNSUPERVISED COREFERENCE RESOLUTION

	MUC			B ³			CEAF		
	ACE-2								
BNEWS	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	68.3	66.6	67.4	70.3	65.3	67.7	–	–	–
A-INF	60.8	61.4	61.1	55.5	69.0	61.5	52.6	52.0	52.3
UNSEL	72.5	65.6	68.9	72.5	66.4	69.3	56.7	64.8	60.5
SUCRE	86.6	60.3	71.0	87.6	64.6	74.4	56.1	81.6	66.5
NWIRE	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	67.7	67.3	67.4	74.7	68.8	71.6	–	–	–
A-INF	62.4	57.4	59.8	59.2	62.4	60.7	46.8	52.5	49.5
UNSEL	76.2	61.5	68.1	81.5	67.6	73.9	61.5	77.1	68.4
SUCRE	82.5	65.7	73.1	85.4	72.3	78.3	63.5	80.6	71.0
NPAPER	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
P&D	69.2	71.7	70.4	70.0	66.5	68.2	–	–	–
A-INF	60.6	56.0	58.2	52.4	60.3	56.0	38.9	44.0	41.3
UNSEL	78.6	65.7	71.6	74.0	68.0	70.9	57.6	73.2	64.5
SUCRE	82.5	67.0	73.9	80.7	69.5	74.6	58.8	77.1	66.7
	ACE2003								
BNEWS	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
H&K	68.3	56.8	62.0	–	–	–	59.9	53.9	56.7
Ng	71.4	56.1	62.8	–	–	–	60.5	53.3	56.7
A-INF	60.9	64.9	62.8	50.9	72.5	59.8	53.8	49.4	51.5
UNSEL	69.5	65.0	67.1	70.2	65.9	68.0	58.5	64.2	61.2
SUCRE	73.9	68.5	71.1	75.4	69.6	72.4	60.1	66.6	63.2
NWIRE	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
H&K	66.2	46.8	54.8	–	–	–	62.8	49.6	55.4
Ng	68.3	47.0	55.7	–	–	–	60.7	49.2	54.4
A-INF	62.7	60.5	61.6	54.8	66.1	59.9	47.7	50.2	49.0
UNSEL	64.8	68.6	66.6	61.5	73.6	67.0	59.8	55.1	57.3
SUCRE	77.6	69.3	73.2	78.8	75.2	76.9	65.1	74.4	69.5

Table 6.9: Scores for MCORE (A-INF, SUCRE and UNSEL) on ACE-2 and ACE2003 and scores for comparable coreference systems.

two motivating examples for our self-trained approach, as follows.

Example 1: better recall effect

Consider the markable pair (“Obama”, “he”) in the test set. Our lexical information was calculated on a version of Wikipedia from 2003 that has no occurrence of “Obama”. Therefore, the N_{MI} score used by A-INF is 0 in this case. However, A-INF is able to find many similar links like (“Clinton”, “he”) and (“Bush”, “he”), links that will get high N_{MI} scores. Suppose we represent links using the following six features: {distance, string match, type of first markable, type of second markable, gender agreement, number agreement}. Then (“Clinton”, “he”), (“Bush”, “he”) and (“Obama”, “he”) will all be assigned the feature vector {1, No, Proper Noun, Personal Pronoun, Yes, Yes}. We can now automatically label Wikipedia using A-INF – this will label (“Clinton”, “he”) and (“Bush”, “he”) as coreferent – and train SUCRE on the resulting training set. SUCRE can then resolve the coreference (“Obama”, “he”) correctly. I call this *better recall effect*.

Example 2: better precision effect

In the scenario of example 1 consider the markable pairs (“Clinton”, “President”) and (“Bush”, “President”), both with a N_{MI} score of more than 0.5, with these feature vectors {8, No, Proper Noun, Proper Noun, Yes, Yes} and {1, No, Proper Noun, Proper Noun, Yes, Yes}. Due to their N_{MI} scores, A-INF incorrectly puts all three markables into one cluster. But as we would expect, A-INF labels many more markable pairs with the second feature vector (distance=1) as coreferent than with the first one (distance=8) in the entire corpus. Now if we train SUCRE on the resulting training set, it can then resolve the similar cases with more precision, (“Clinton”, “President”) as disreferent and (“Bush”, “President”) as coreferent. I call this *better precision effect*.

Turning now to the experimental evidence in table 6.9, it can be seen that recall and precision of UNSEL are consistently better than A-INF. These results provide suggestive evidence for the *better precision effect* and *better recall effect* that were explained above.

Briefly, the advantages of our self-training approach are: (i) cases can be covered that do not occur in the unlabeled corpus (better recall effect); and

Gold setting + True markables				
System	MD	MUC	B ³	CEAF
Relax	100	33.7	84.5	75.6
SUCRE-2010	100	60.8	82.4	74.3
SUCRE	100	64.3	87.0	80.1
UNSEL	100	63.0	86.9	79.7
Automatic setting + System markables				
System	MD	MUC	B ³	CEAF
Tanl-1	73.9	24.6	61.3	57.3
SUCRE-2010	80.7	52.5	67.1	62.7
SUCRE	80.9	55.7	69.7	66.6
UNSEL	80.9	55.0	69.8	66.3

Table 6.10: F_1 -Scores for MSCORE (SUCRE and UNSEL) on OntoNotes and F_1 -Scores for comparable coreference systems from SemEval-2010. MD: Markable Detection F-Score [Recasens et al., 2010].

(ii) the leveraging effect of fine-grained linguistic features like distance, person, number and gender can be used to improve precision (better precision effect).

Of course, a “deterministic” system (Haghighi and Klein [2009] Raghunathan et al. [2010]) would also resolve this type of coreference correctly, but as was argued in section 6.5.1, there are advantages to a learned system, especially one that can be learned without using data labeled by humans.

Supervised vs. Unsupervised

Table 6.10 compares our supervised (SUCRE) and unsupervised self-trained (UNSEL) models with the recently published SemEval-2010 OntoNotes results (gold and automatic settings). I compare with the scores of the two best systems, Relax and SUCRE-2010 (for the gold setting with true markables, i.e. 100% markable detection (MD) F-Score) and Tanl-1 and SUCRE-2010 (for the automatic setting with system markables, 89.9% markable detection (MD) F-Score). It is apparent from this table that our supervised and unsupervised self-trained models outperform both Tanl-1 and SUCRE-2010. I should make clear that I did not use the test set for development in order to ensure a fair comparison with the participant

systems at SemEval-2010.

The comparison of SUCRE and UNSEL (tables 6.9 and 6.10) shows that the unsupervised self-trained system (UNSEL) does a lot worse than the supervised system (SUCRE) on ACE (table 6.9). In contrast, UNSEL performs almost as well as SUCRE on OntoNotes (table 6.10), for both gold and automatic settings: F_1 differences range from +0.1 (Automatic, B³) to -1.3 (Gold, MUC). A possible explanation for this might be that the approaches of OntoNotes and ACE to the true markables are different. The true markables in OntoNotes include all markables that are singletons, where a singleton is defined as a markable that is not part of a coreference chain. Very few of the singletons in ACE were included in the true markables. In any real scenario, a large proportion of markables will be singletons and any real system has to perform markable detection without access to true markables. Our results show that in a realistic scenario (OntoNotes: automatic setting and system markables), using lexical information in an unsupervised self-trained system is almost as good as a system trained on manually-labeled data.

6.6 Conclusion

In this chapter, I first introduced a feature space model of natural language that contains two feature spaces: shallow space, with only one term-frequency-based feature; and rich space, which is n -dimensional. For the shallow feature space, a case study was presented which is itself a novel method for word alignment.

I also showed that lexical information can be used to develop an unsupervised model for shallow coreference resolution (i.e., in shallow feature space), where only lexical information is used (i.e., association scores) for finding coreference chains. Then an unsupervised self-trained approach was introduced, where a supervised model is trained on a corpus that was automatically labeled by the unsupervised model based on lexical information. The results of the experiments indicate that the performance of the unsupervised self-trained approach is better than the performance of other unsupervised learning systems. In addition, I showed that my proposed system is a flexible and modular framework that is able to learn from data with different quality and domain. Not only is it able to deal with shallow information spaces, but it can also deliver competitive results

CHAPTER 6. UNSUPERVISED COREFERENCE RESOLUTION

for rich feature spaces. In addition, our framework is the first CoRe system that is designed to support the three main modes of machine learning equally well: supervised, self-trained (semi-supervised) and unsupervised.

Chapter 7

Conclusions and Future Work

7.1 Contributions of this Thesis

The major contributions of this thesis are as follows:

- In **Chapter 4**, a modular supervised system for coreference resolution was presented. It is composed of separate and interchangeable components which were designed by breaking down system into modules. This framework reduces the time and effort needed for creating a coreference resolution system. In this approach each module accomplishes one function and contains everything necessary to accomplish this; this improves the modularity. It also improves maintainability by enforcing logical boundaries between components.
- In **Chapter 5**, a novel approach for feature engineering of natural language processing was presented that is based on representing unstructured text in a relational data model. This approach provides more coherency of the methods and algorithms that are needed in different stages by presenting a common structure of the text corpus based on the relations between the textual entities. It is a natural formalism for supporting the definition and extraction of features. It can be used to reduce the size of the feature vector (in the feature space of an NLP task). The modular architecture of our framework makes it possible to use it with other machine learning tools. It is possible to apply many existing methods of knowledge discovery

in database to the text corpus with less design and implementation effort.

- In **Chapter 6**, a new framework was presented that supports three major modes of machine learning, namely, (i) Supervised; (ii) Unsupervised; and (iii) Self-Trained. First, it was demonstrated that lexical information can be used to develop an unsupervised model for shallow coreference resolution. Then, an unsupervised self-trained method was introduced that takes a two-learner two-feature-space approach. The two learners are supervised and unsupervised. The feature spaces are the shallow and rich feature spaces. It was also shown that the performance of the unsupervised self-trained system is better than the performance of other unsupervised systems when it is self-trained on the automatically labeled corpus and uses the leveraging effect of rich linguistic features. Our framework is a flexible and modular framework that is able to learn from data of different quality and domain. Not only is it able to deal with shallow information spaces, but it can also deliver competitive results for rich feature spaces.

7.2 Future Work

Future work can be done in the following directions:

7.2.1 Chain-based coreference resolution

Here, a set of hard linguistic constraints for chains needs to be developed. An example of a hard constraint is that a chain in English must not contain two markables with conflicting semantic gender. Given the importance of linguistic filtering for the success of pair-based coreference resolution, chain-based hard constraints are clearly a necessary first step in chain-based coreference resolution.

For the chain-based model, a log-linear model will be used for the distribution P on partitions π from which samples are drawn (w_i are weight parameters, f_i are global features). This is defined as follows:

$$P(\pi) = \frac{\exp(\sum_i w_i f_i(\pi))}{\sum_{\pi'} \exp(\sum_i w_i f_i(\pi'))} \quad (7.1)$$

π is the set of all partitions of a document. The space of all partitions is searched by moving from one partition to the next until an optimum is reached. The

transitions are performed according to a binary decision: move or stay. The transitions are continued until an optimal partition is found (i.e. *stop-condition*: staying at a partition for a complete pass through all markables of the document).

If we are at partition π_1 and want to transit to partition π_2 , there are these possibilities:

$$P(\pi_1) < P(\pi_2) \tag{7.2}$$

$$P(\pi_1) > P(\pi_2) \tag{7.3}$$

$$P(\pi_1) = P(\pi_2) \tag{7.4}$$

Then a transit from π_1 to π_2 is desired if the condition of equation 7.2 is true. From equation 7.2 we have:

$$\frac{P(\pi_1)}{P(\pi_2)} < 1 \tag{7.5}$$

From equation 7.1:

$$\frac{P(\pi_1)}{P(\pi_2)} = \frac{e^{\sum_i w_i f_i(\pi_1)}}{e^{\sum_i w_i f_i(\pi_2)}} = e^{\sum_i w_i (f_i(\pi_1) - f_i(\pi_2))} \tag{7.6}$$

and from equation 7.5:

$$e^{\sum_i w_i (f_i(\pi_1) - f_i(\pi_2))} < 1 \tag{7.7}$$

$$\sum_i w_i (f_i(\pi_1) - f_i(\pi_2)) < 0 \tag{7.8}$$

Equation 7.8 suggests that our problem is a binary classification that can be solved using a feature space of the form $f_i(\pi_1) - f_i(\pi_2)$ and two classes *Transit* (*value* < 0) and *Stay* (*value* >= 0). We would then sample from the automatically labeled data and train a decision tree.

The equation suggests that the problem is a binary classification task that can be resolved using a feature space of the form $f_i(\pi_1) - f_i(\pi_2)$ and two classes: *move* ($\vec{w}(\vec{\pi}_1 - \vec{\pi}_2) < 0$) and *stay* ($\vec{w}(\vec{\pi}_1 - \vec{\pi}_2) \geq 0$). We would sample from the automatically labeled data, as described in figure 7.1 and train a decision tree. For decoding (finding the best partition), we would also start from a partition of all singleton clusters, then proceed through the markables starting from the second markable of the document. The transition from one partition to the next would then be repeated until no further improvement is possible.

DoSampling (M_1, M_2, \dots, M_n)

1. Build partition π_1 : for each markable M_i : $C_i \leftarrow \{M_i\}$
2. Set current partition π_c equal to π_1 .
3. Start from the second markable of the document: $j \leftarrow 2$
4. For each cluster C_i preceding M_j :
 - (a) build π_n by merging C_i and C_j : $C_i \leftarrow C_i \cup C_j$
 - (b) **If** M_j belongs to C_i (i.e. coreferent):
 - i. add the transition $\pi_c \rightarrow \pi_n$ as a positive instance;
 - ii. set current partition π_c equal to π_n .
 - iii. go to 5
 - Else:** add the transition $\pi_c \rightarrow \pi_n$ as a negative instance
 - (c) $j \leftarrow j + 1$.
5. Stop if $j = n$ otherwise $j \leftarrow j + 1$ and go to 4

Figure 7.1: Sampling algorithm.

This method is more efficient than the chain-based coreference algorithms that have been proposed in the literature. However, it only searches a sub-part of the entire search space and does not necessarily find the global optimum. To investigate to what extent this is a problem, starting partitions that are different from the “singleton partition” in figure 7.1 could be used. These starting partitions could be assembled by merging markables with chains, where features are defined on the markable-chain pair only, not on the entire partition. This approach is more similar to our pair-based work, where the coreference probability of pairs of two markables is estimated and the resulting set of probability estimates is then used to assemble chains in a way that is standard in coreference resolution (see description of SUCRE in chapter 4).

7.2.2 Self-training for chains

For chain-based self-training, it is necessary to find chain-based equivalents of the method which was defined in chapter 6. First, the mutual information measure that we used for markable pairs needs to be generalized *to markable sets*. Experiments could be performed with *co-information* (CI) and *pointwise co-information* (PCI), generalizations of mutual information and pointwise mutual information to more than two random variables.

The CI of a set of markables is the amount of information shared by all markables in the set. CI for $V = (x_1, x_2, \dots, x_n)$ is defined as follows [Bell, 2003]:

$$\text{CI}_n(V) = - \sum_{U \subseteq V} (-1)^{|U|} H(U)$$

For example, for three markables the value of CI is the following:

$$\text{CI}_3(a, b, c) = \sum_{i \in \{\bar{a}, a\}} \sum_{j \in \{\bar{b}, b\}} \sum_{k \in \{\bar{c}, c\}} P(i, j, k) \log \frac{P(i, j)P(j, k)P(i, k)}{P(i, j, k)P(i)P(j)P(k)}$$

To compute CI_n , we need to sum up 2^n summands (entropies). As a measure that can be computed more efficiently, $\text{PCI}_n(V)$ is defined as the equivalent to the single summand of CI that only considers positive information:

$$\text{PCI}_n(V) = \log \frac{\prod_{U \subseteq V, |U| \bmod 2=0} P(U)}{\prod_{T \subseteq V, |T| \bmod 2=1} P(T)}$$

Note the analogy to pointwise mutual information, which is similarly defined as the positive summand that enters in the computation of mutual information. This analogy becomes clearer when we look at the instantiation of PCI_n for a particular n , e.g., $n = 3$:

$$\text{PCI}_3(a, b, c) = \log \frac{P(a, b)P(b, c)P(a, c)}{P(a, b, c)P(a)P(b)P(c)}$$

$\text{PCI}_3(a, b, c)$ is one of the 2^3 summands whose weighted sum is $\text{CI}_3(a, b, c)$.

CI and PCI scores are used for computation of a partition based on CI and PCI scores as distance measures (or rather similarity measures) between two sets of markables or chains. Let us call the information-theoretic similarity measure we are using IM, where $\text{IM} \in \{\text{CI}, \text{PCI}\}$. First each markable M_i is assigned to its own chain C_i . Then steps 2 and 3 are repeated:

1. Chain size $k \leftarrow 2$
2. For all subsets \mathcal{C} of chains with $|\bigcup \mathcal{C}| = k$, pick the one that has the largest value $\text{IM}_k(\bigcup \mathcal{C})$
3. If $\text{IM}_k(\bigcup \mathcal{C}) > \theta_k$, then merge the chains in \mathcal{C} into a new chain and go to step 2. Otherwise $k \leftarrow k + 1$. If k is less than the number of markables, go to step 2, otherwise stop.

Initially, the algorithm can be tested for up to $k = 3$, due to the complexity of this method, but the higher values should be tested if the first experiments indicate that higher values will improve performance.

7.3 Summary

Chapter 2 presented an overview of the field of machine learning by describing its foundational background and reviewing the basics of different approaches to it. This Chapter laid down the necessary machine learning foundations related to our proposed methods, systems and framework.

Chapter 3 reviewed the literature and related work concerning coreference resolution. I reviewed the existing coreference resolution methods in four categories: (i) Rule-based approaches, where Hobbs [Hobbs, 1986], CogNIAC [Baldwin, 1996],

Mitkov [Mitkov, 1998] and Multi-Pass-Sieve [Raghunathan et al., 2010] methods were presented; (ii) Supervised approaches, where the decision tree [Soon et al., 2001], generative probabilistic [Ge et al., 1998] and ranking approaches [Denis and Baldridge, 2007a] were shown; (iii) Unsupervised approaches, where a nonparametric Bayesian model [Haghighi and Klein, 2007] and a Markov logic model [Poon and Domingos, 2008] were presented; and (iv) Semi-supervised approaches, where Self-Training [Ng and Cardie, 2003] and Co-Training [Müller et al., 2002] methods were presented. The most commonly used metrics for evaluation of coreference resolution were also presented, accompanied with a running example. We showed that it is insufficient to report only on one of the above mentioned metrics.

In Chapter 4, I presented SUCRE, a modular supervised system for coreference resolution. When compared with existing systems, the most important advantage of our system is its modularity and its flexible method of feature engineering. I reported on the successful participation of SUCRE in SemEval-2010 Task 1 on Coreference Resolution in Multiple Languages [Recasens et al., 2010] and in CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNote [Pradhan et al., 2011b]. In SemEval-2010, I obtained the best results in several categories, including the regular closed annotation tracks of English, German and Italian.

Chapter 5 presented a new framework for feature engineering of natural language processing that is based on representing unstructured text in a relational database model. It includes powerful and flexible methods for implementing and extracting new features. It allows for systematic and fast searches of the space of features, thereby reducing the time and effort needed for creating an NLP system. The relational database model at the heart of our framework makes it possible to use SQL as the feature definition language. However, as SQL could make the system cumbersome to use, a pseudo-language was also defined which is syntactically more convenient. An example application of our framework was also presented that deals with the problem of coreference resolution. The results of the system in multiple languages show that our approach to feature engineering supports rapid and high-quality development of a machine-learning based NLP system. A comparative study of coreference resolution features across different languages was also presented. The important distinction between *identical*, *uni-*

versal and *language-specific* features was also introduced and the first two groups were investigated for four languages, Dutch, German, Italian and Spanish.

In Chapter 6, I first introduced a feature space model of natural language that contains two feature spaces: shallow space, with only one term-frequency-based features; and rich space that is n-dimensional. For the shallow feature space, I presented a case study which by itself is a novel method for word alignment. It was also shown that lexical information can be used to develop an unsupervised model for shallow coreference resolution (i.e., in shallow feature space), where only lexical information (i.e., association scores) is used for finding coreference chains. Then I introduced an unsupervised self-trained approach where a supervised model is trained on a corpus that was automatically labeled by the unsupervised model based on lexical information. The results of the experiments indicate that the performance of the unsupervised self-trained approach is better than the performance of other unsupervised learning systems. In addition, we showed that our system is a flexible and modular framework that is able to learn from data with different quality and domain. Not only is it able to deal with shallow information spaces, but it can also deliver competitive results for rich feature spaces. In addition, our framework is the first coreference resolution system that is designed to support the three main modes of machine learning equally well: supervised, self-trained (semi-supervised) and unsupervised.

Appendix A

Visualization

A.1 Introduction

Here a visualization method of coreference data will be presented based on Self Organizing Maps (SOMs) [Burkovski et al., 2010]. This is a joint work with Andre Burkovski¹ from the Visualization and Interactive Systems Group at the University of Stuttgart. This work was published in the technical report No. 2010/09 at the faculty of Computer Science, Electrical Engineering, and Information Technology, University of Stuttgart.

One of our goals is to enable a researcher to explore the feature space used for machine learning. This can aid in seeing areas in the data where coreferent data is not clearly separable from other data. The user is able to utilize the information for the design of new features, thereby better solving a specific problem. A SOM is a type of artificial neural network which projects high-dimensional input data on a low dimensional map. Thus, it is suitable as a basis for visualizations of high dimensional coreference feature space. The benefit of using SOM based visualizations for three applications regarding coreference resolution will be discussed. The first application is the presentation of high dimensional coreference data and their features in a low dimensional space. This allows a better understanding of the data distribution in the feature space. The second application is the design of new features based on knowledge gathered through SOM based data

¹The visualization part of this work was done by Andre Burkovski. I did the coreference part.

exploration. The third application is an annotation task, where a user can annotate data with coreference information. This application shows that SOM based visualizations are capable of reducing the time and effort needed for annotating large documents.

A.2 Method

As mentioned above, we use self organizing maps (SOMs) [Kohonen, 1982] for visualizing coreference. SOMs are a type of artificial neural network that maps high dimensional data to a topology preserving representation of typically two or three dimensions. We have developed an interactive visualization tool based on SOMs for CoRe annotation.

The visualizations provided by the Matlab SOM-Toolbox [Vesanto et al., 1999] are static and do not facilitate data exploration. Therefore, an interactive visualization was developed based on the well-known *unified distance matrix* (U-matrix) [Ultsch and Siemon, 1990]. The U-matrix is intended as an intuitive representation of distances between nodes, with high U-matrix values indicating large distances between neighboring nodes in feature space. A common approach to visualize the U-matrix is to display cells for both nodes and edges. In contrast, our visualization treats the U-matrix as a graph. In figure A.1 (a), a U-matrix for the first ten documents of the MUC-6 corpus can be seen. Rectangles indicate nodes of the SOM, and the edges represent the neighborhood relations of the nodes according to the map topology. Large distances between neighboring nodes are depicted in red; small distances, in blue. Bright blue areas are clusters; the lower right (except for the single green cluster at the corner of the diagram) is an example. The size and number of each node indicates the number of input vectors which were projected onto this node. Displaying the numbers in this fashion is referred to as a hit histogram in the SOM literature and is an indication of the density of different areas of the SOM map. The user may examine individual map units and their corresponding weight vectors. Users are also able to inspect the links assigned to a node and display coreference information. Thus, this U-matrix visualization gives an overview of the distance structure of the map in feature space, which allows the identification of clusters.

An alternative for coloring nodes with U-matrix values are *component planes*

APPENDIX A. VISUALIZATION

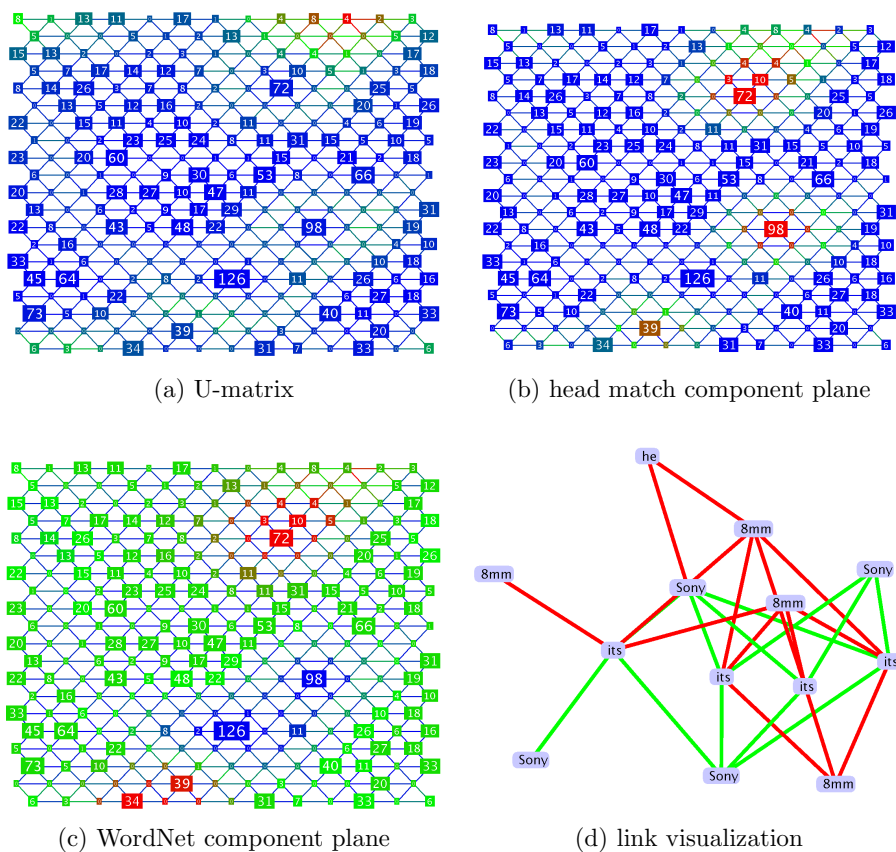


Figure A.1: A subset of the visualizations we have developed. See text for discussion.

(CPs). Colors correspond to the weight vector value of a node for a single feature vector component. These CPs allow the visualization of the influence of a single feature on the cluster formation. Figure A.1 (b) and figure A.1 (c) show two examples. A high influence of the feature on the projection of data vectors is displayed in red; low influence, in blue. Figure A.1 (b) shows the weight of the head-match feature. Since this feature is a strong coreference indicator, one can expect that links assigned to SOM nodes with high influence are mostly coreferent. Figure A.1 (c) shows the influence of the WordNet distance. A high influence indicates that markables of links in this node have a high similarity. Blue nodes

contain links where no WordNet definition exists. Green nodes contain links that are present in WordNet, but with low similarity. Browsing through component planes is an important technique for users who want to explore the feature space and the influence of features on coreference resolution. It is also critical for successful bulk labeling. A strong indicator for coreference, like head match or WordNet, helps in recognizing homogeneous clusters of coreferent markable pairs. The user can then accurately annotate entire nodes of data.

Users can switch between component planes and apply different SOM visualization methods. One method is based on force-directed graph layout algorithms [Fruchterman and Reingold, 1991, Eades, 1984]. This method rearranges nodes of the SOM according to their distances. The edges represent the springs and forces in the force-directed model. A further visualization displays connections between the first and second best matching units (BMU) of feature vectors. The visualization is based on the work of [Tasdemir and Merényi, 2009]. This visualization helps users identify well clustered areas in the SOM. See [Burkovski et al., 2010] for details.

A.2.1 Interaction techniques

When a SOM visualizes large amounts of data, each SOM node contains a large number of markable pairs. Thus, tools are also needed for detailed inspection of the data at a particular SOM node or in a small area of the SOM map. For this purpose, we have developed a number of interaction techniques.

The simplest interaction technique allows users to inspect and label individual markable pairs at a SOM node. Users can also look at the textual context of a particular markable pair, and select several nodes and label the contents with coreference information and a confidence level for possibly mixed nodes. Such data is important for machine learning methods that can take advantage of labels that are provided with a confidence level.

Another interaction technique is the *SOM-zoom*, which is related to a technique used in hierarchical SOMs [Rauber et al., 2002]. In hierarchical SOMs, only a small map is trained, which is subsequently refined by training a new SOM for each of its nodes. The refinement is iterated until a desired level of hierarchy is reached. We adapted this method to better suit the data exploration task and,

therefore, to use only data of nodes selected by the user. The user can train a new SOM on arbitrary subsets of map nodes to gain detailed insight into the structure of the data.

A third interaction technique is the *coreference visualization graph* (CVG). The basic motivation is fast decision making on the coreference status of the links in a SOM node. In our experiments we observed that many links in a SOM node are connected with each other by a common markable. The connection depends on the link generation strategy in the preprocessing phase and the transitivity of links.

The idea of the visualization is to create graphs of connected links, where every node is a markable and an edge representing a link between markables. Thus transitive links can be visualized via a connected graph. We experimented with a micelle form and a graph form of the relation. Early on, the graph form proved to be more efficient.

We developed several interaction techniques for link annotation in CVG. The user can select (i) a single edge (which represents a link); (ii) a single node (which selects each incoming edge and therefore all links connected with a markable); (iii) multiple nodes; and (iv) multiple graphs. After selection, the user can further inspect the links or label them. Figure A.1 (d) shows a coreference graph after the user has performed a number of labeling actions. Red edges represent disreferent links; green edges, coreferent links. Each markable is labeled with its head word.

A.2.2 Annotation

During the annotation process, users can interact with the SOM by selecting one or several nodes. They can start a new SOM calculation or load an already existing one. Users can switch freely back and forth between component planes and display additional information. The additional information may include gold standard information (if it is available), labeling progress, and coreference and disreference proportions. As was just discussed, users can inspect individual SOM nodes or small areas of the SOM in detail by using different interaction techniques to see textual context, a SOM of a subset of nodes, and CVG displays.

We tested the prototype in an annotation user study. The results indicate that the tool is able to produce more links than other tools and that the created

samples are of higher quality.

A.2.3 Feature engineering

We also tested the utility of our visualizations for feature engineering applications, in particular for feature design, analysis and modification. The main supporting functionality provided by the visualization for feature engineering is the identification of heterogeneous clusters. Such clusters occur because feature representations of a group of markable pairs are similar, but some members of the group are coreferent and others disreferent.

Usually, heterogeneous data clusters are created by the influence of one or more features. Features that have a high influence on cluster formation can be found using component planes like the ones shown in figure A.1 (b) and (c). Feature space exploration via the component planes enables users to identify the critical features for a cluster.

The user is able to gain a deeper understanding of a heterogeneous cluster by looking at example markable pairs and trying to devise features that would separate coreferent from disreferent links. The clustering performance of the SOM can then be improved using the additional features. The new features are also beneficial for more effective and faster annotation.

Heterogeneous clusters can also be analyzed by the SOM-zoom functionality and by experimenting with different feature subsets. This is especially promising for heterogeneous clusters with high variance in their feature vectors. Although it did not deliver new insights in initial experiments, the functionality may prove useful with the improvement of features in the future. The recalculation of a new SOM for mixed nodes and a subset of features used may result in a different, better ordering of the links.

Although the interfaces developed for annotation were partially usable for feature engineering, they were not designed with this purpose in mind, and are in many respects not ideal for the expert user who wants to understand system behavior (as opposed to performing labeling).

We developed the visual interfaces for annotation, but they are also partially usable for feature engineering. However, we realized that expert users require more powerful methods and tools to understand system behavior than human

annotators. Based on this experience, we have concluded that it is difficult to develop a CoRe visualization system that works equally well for human annotators and NLP developers.

In human annotation, the goal is to present all information relevant for making correct and efficient annotation decisions. In contrast, the NLP developer needs to understand the impact that features, parameters and system configurations have on CoRe performance. While systems optimized for these two different applications can share a number of components, there are also several requirements that are fundamentally different.

In addition to the U-matrix-based SOM visualizations in figure A.1, we also experimented with a number of alternatives: the P-matrix (a probability density estimation in a map unit), the U*-matrix (a combination of U-matrix and P-matrix), distribution of weight influence (a bar or pie chart for a map unit's weight values similar to component planes), and PCA projection of the map. We did not find any advantages compared to the U-matrix for annotation and feature engineering.

A.3 Summary

Here we have presented visualization techniques for our interactive user interface for the exploration and annotation of coreference information. It uses Self Organizing Maps (SOM) to create a low-dimensional representation of high-dimensional feature data.

First, the low dimensional presentation space of the SOM enables the user to explore the high dimensional feature space. The user is better able to understand how the data is organized and why (interaction techniques). The second application enables the user to annotate many links at once. It also allows the assignment of user-specified confidence values to labels. Third, the SOM allows the user to judge the quality of the features and to explore nodes with mixed coreference information. The content of these nodes can help the user design new features.

Appendix B

SemEval-2010 Feature Sets

B.1 Catalan

The feature set used for Catalan is as follows:

1. If the markables are in the same sentence, return 1; otherwise 0.
2. If the distance between the markables is one sentence, return 1; otherwise 0.
3. If the distance between the markables is more than one sentence, return 1; otherwise 0.
4. If one markable head word is the acronym¹ of another one, return 1; otherwise return 0.
5. If the named entity attribute of both markables are known and equal, return 1; otherwise return 0.
6. If there is case-sensitive head word matching, return 1; otherwise 0.
7. If there is case-insensitive head word sub-string matching, return 1; otherwise 0.

¹A word formed from the initial letters or groups of letters of words in a set phrase or series of words.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

8. If the head word of both markables are proper nouns, then return 1; otherwise return 0.
9. If the head word of the first markable is a proper noun and the head word of the second markable is a common noun, then return 1; otherwise return 0.
10. If the head word of the first markable is a common noun and the head word of the second markable is a proper noun, then return 1; otherwise return 0.
11. If the head word of both markables are common nouns, then return 1; otherwise return 0.
12. If the first markable starts with an indefinite article, then return 1; otherwise return 0.
13. If the pronoun type of the first markable is possessive and both markables match in number, return 1; otherwise 0.
14. If the pronoun type of the first markable is demonstrative and both markables match in number, return 1; otherwise 0.
15. If the pronoun type of the first markable is relative and both markables match in number, return 1; otherwise 0.
16. If the pronoun type of the first markable is personal and both markables match in number, return 1; otherwise 0.
17. If the pronoun type of the second markable is possessive and both markables match in number, return 1; otherwise 0.
18. If the pronoun type of the second markable is demonstrative and both markables match in number, return 1; otherwise 0.
19. If the pronoun type of the second markable is relative and both markables match in number, return 1; otherwise 0.
20. If the pronoun type of the second markable is personal and both markables match in number, return 1; otherwise 0.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

21. If the markables are connected to the same word in the dependence parse structure, return 1; otherwise 0.
22. If the head of one markable is directly connected to the head of another one in the dependency parse structure, return 1; otherwise 0.
23. If both markables have subject role, return 1; otherwise return 0.
24. If none of the markables have subject role, return 1; otherwise return 0.
25. If the head of the first markable is a subject and the head of the second markable is head of a noun phrase, return 1; otherwise 0.
26. If the head of the first markable is a subject and the head of the second markable is a specifier, return 1; otherwise 0.
27. If the head of the second markable is a subject and the head of the first markable is a specifier, return 1; otherwise 0.
28. If the head of the first markable is a subject and the head of the second markable is a direct object, return 1; otherwise 0.
29. If the head of the first markable is a subject and the head of the second markable is an indirect object, return 1; otherwise 0.
30. If the head of the first markable is head of a noun phrase and the head of the second markable is a specifier, return 1; otherwise 0.
31. If the head of the second markable is head of a noun phrase and the head of the first markable is a specifier, return 1; otherwise 0.
32. If the head of the first markable is a specifier and the head of the second markable is a specifier, return 1; otherwise 0.
33. If the markables' head words agree in number, then return 1; otherwise 0.
34. If the markables' head words agree in gender, then return 1; otherwise 0.
35. If both markables are pronouns and are of the same type, then return 1; otherwise 0.

36. If both markables are pronouns and agree in person, then return 1; otherwise 0.
37. If both markables agree in case (i.e. accusative, dative, nominative and oblique), then return 1; otherwise 0.

B.2 German

The features used for German are listed as follows:

1. If two markables are in the same sentence return 1; if the distance is one sentence, return 2; if the distance is more than one sentence return 3.
2. If one markable head word is the acronym of another one, return 1; otherwise return 0.
3. If the named entity attributes of both markables are known and equal, return 1; otherwise return 0.
4. If there is head word string matching, return 1; otherwise 0.
5. If there is case-insensitive head word sub-string matching, return 1; otherwise 0.
6. If the first word of the second markable contains string “ein” and is an article, then return 1; otherwise return 0.
7. If one markable includes another one, return 1; otherwise return 0.
8. If one markable overlaps another one, return 1; otherwise return 0.
9. If the head of the first markable is a proper noun and the head of second one is a proper noun, return 1; otherwise return 0.
10. If the head of the first markable is a proper noun and the head of second one is a common noun, return 1; otherwise return 0.
11. If the head of the first markable is a common noun and the head of second one is a proper noun, return 1; otherwise return 0.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

12. If the head of the first markable is a common noun and the head of second one is a common noun, return 1; otherwise return 0.
13. If both markables have subject role, return 1; otherwise return 0.
14. If none of the markables have subject role, return 1; otherwise return 0.
15. If the first markable is an attributive demonstrative pronoun and the markables agree in number, then return 1; otherwise return 0.
16. If the first markable is a substitutive demonstrative pronoun and the markables agree in number, then return 1; otherwise return 0.
17. If the first markable is an attributive indefinite pronoun and the markables agree in number, then return 1; otherwise return 0.
18. If the first markable is a substitutive indefinite pronoun and the markables agree in number, then return 1; otherwise return 0.
19. If the first markable is a personal pronoun and the markables agree in number, then return 1; otherwise return 0.
20. If the first markable is a reflexive pronoun and the markables agree in number, then return 1; otherwise return 0.
21. If the first markable is a substituting possessive pronoun and the markables agree in number, then return 1; otherwise return 0.
22. If the first markable is an attributive possessive pronoun and the markables agree in number, then return 1; otherwise return 0.
23. If the first markable is an attributive relative pronoun and the markables agree in number, then return 1; otherwise return 0.
24. If the first markable is a substitutive relative pronoun and the markables agree in number, then return 1; otherwise return 0.
25. If the second markable is an attributive demonstrative pronoun and the markables agree in number, then return 1; otherwise return 0.

26. If the second markable is a substitutive demonstrative pronoun and the markables agree in number, then return 1; otherwise return 0.
27. If the second markable is an attributive indefinite pronoun and the markables agree in number, then return 1; otherwise return 0.
28. If the second markable is a substitutive indefinite pronoun and the markables agree in number, then return 1; otherwise return 0.
29. If the second markable is a personal pronoun and the markables agree in number, then return 1; otherwise return 0.
30. If the second markable is a reflexive pronoun and the markables agree in number, then return 1; otherwise return 0.
31. If the second markable is a substitutive possessive pronoun and the markables agree in number, then return 1; otherwise return 0.
32. If the second markable is an attributive possessive pronoun and the markables agree in number, then return 1; otherwise return 0.
33. If the second markable is an attributive relative pronoun and the markables agree in number, then return 1; otherwise return 0.
34. If the second markable is a substitutive relative pronoun and the markables agree in number, then return 1; otherwise return 0.
35. If the markables' head words agree in number and their numbers are known, then return 1; otherwise 0.
36. If the markables' head words agree in grammatical gender and their genders are known, then return 1; otherwise 0.
37. If the markables' head words agree in case (i.e. accusative, dative, nominative and genitive) and their cases are known, then return 1; otherwise 0.
38. If the markables' head words agree in pronoun person (i.e. first, second or third) and their pronoun persons are known, then return 1; otherwise 0.

B.3 English

The feature set of English consists of 27 features:

1. If two markables are in the same sentence return 1; if the distance is one sentence, return 2; if the distance is more than one sentence return 3.
2. If one markable head word is the acronym of another one, return 1; otherwise return 0.
3. If there is case-insensitive head word matching, and both markables are proper nouns, then return 1; otherwise return 0.
4. If there is case-insensitive head word matching, and both markables are pronouns, then return 1; otherwise return 0.
5. If there is case-insensitive head word matching, and both markables are common nouns, then return 1; otherwise return 0.
6. If there is no head word string matching, but the head word of the first markable matches any word of the second markable, and the second markable is a proper noun, then return 1; otherwise return 0.
7. If there is no head word string matching, but the head word of the second markable matches any word of the first markable, and the first markable is a proper noun, then return 1; otherwise return 0.
8. If there is no head word string matching, but a word in the second markable matches a word in the first markable, and both markables are proper nouns, then return 1; otherwise return 0.
9. If the markables' head words agree in number and their numbers are known, then return 1; otherwise 0.
10. If the beginning words of the markables agree in number and their numbers are known, then return 1; otherwise 0.
11. If the Animacy semantic class (i.e. person vs. not person) of the head words are known and match, then return 1; otherwise 0.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

12. If the pronoun persons (i.e. first, second and third) of the head words are known and match, then return 1; otherwise 0.
13. If the first word of the second markable is equal to (case-insensitive) “a”, “an”, “some” or “any”, then return 1; otherwise 0.
14. If the first word of the second markable is equal to (case-insensitive) “this”, “that”, “these” or “those”, then return 1; otherwise 0.
15. If the pronoun type of the first markable is known and is possessive, return 1; otherwise 0.
16. If the pronoun type of the first markable is known and is determiner, return 1; otherwise 0.
17. If the pronoun type of the first markable is known and is reflexive, return 1; otherwise 0.
18. If the pronoun type of the first markable is known and is subjective, return 1; otherwise 0.
19. If the pronoun type of the first markable is known and is objective, return 1; otherwise 0.
20. If the pronoun type of the second markable is known and is possessive, return 1; otherwise 0.
21. If the pronoun type of the second markable is known and is determiner, return 1; otherwise 0.
22. If the pronoun type of the second markable is known and is reflexive, return 1; otherwise 0.
23. If the pronoun type of the second markable is known and is subjective, return 1; otherwise 0.
24. If the pronoun type of the second markable is known and is objective, return 1; otherwise 0.
25. If both markables have subject role, return 1; otherwise return 0.

26. If none of the markables have subject role, return 1; otherwise return 0.
27. If the markables are directly connected in the dependence parse structure, return 1; otherwise return 0.

B.4 Spanish

The feature set of Spanish is identical to the feature set of Catalan (see section B.1).

B.5 Italian

The feature set used for Italian contains the following features:

1. If the markables in the same sentence, return 1; otherwise 0.
2. If the distance between the markables is one sentence, return 1; otherwise 0.
3. If the distance between the markables is more than one sentence, return 1; otherwise 0.
4. If one markable head word is the acronym of another one, return 1; otherwise return 0.
5. If both markables agree in number and their named entity attributes are known and equal, return 1; otherwise return 0.
6. If the named entity attribute of both markables are known and not equal, return 1; otherwise return 0.
7. If there is case-sensitive head word string matching, and both markables are nouns, then return 1; otherwise return 0.
8. If there is case-insensitive head word sub-string matching, and both markables are nouns, then return 1; otherwise return 0.

9. If there is case-sensitive head word string matching, and both markables are pronouns, then return 1; otherwise return 0.
10. If there is case-insensitive head word sub-string matching, and both markables are pronouns, then return 1; otherwise return 0.
11. If the markables' head words agree in number and their numbers are known, then return 1; otherwise 0.
12. If one markable includes another one, return 1; otherwise return 0.
13. If one markable overlaps another one, return 1; otherwise return 0.
14. If the head of the first markable is a noun and the head of second one is a noun, return 1; otherwise return 0.
15. If the head of the first markable is a noun and the head of second one is a pronoun, return 1; otherwise return 0.
16. If the head of the first markable is a pronoun and the head of second one is a noun, return 1; otherwise return 0.
17. If the head of the first markable is a pronoun and the head of second one is a pronoun, return 1; otherwise return 0.
18. If both markables have subject role, return 1; otherwise return 0.
19. If none of the markables have subject role, return 1; otherwise return 0.
20. If the first word of the second markable contains string "un" and is an article, then return 1; otherwise return 0.

B.6 Dutch

The feature set of Dutch is as follows:

1. If the markables are in the same sentence, return 1; otherwise 0.
2. If the distance between the markables is one sentence, return 1; otherwise 0.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

3. If the distance between the markables is more than one sentence, return 1; otherwise 0.
4. If one markable head word is the acronym of another one, return 1; otherwise return 0.
5. If there is a case-sensitive head word string matching, and both markables are nouns, then return 1; otherwise return 0.
6. If there is a case-insensitive head word sub-string matching, and both markables are nouns, then return 1; otherwise return 0.
7. If there is a case-sensitive head word string matching, and both markables are pronouns, then return 1; otherwise return 0.
8. If there is a case-insensitive head word sub-string matching, and both markables are pronouns, then return 1; otherwise return 0.
9. If the head of the first markable is a noun and the head of second one is a noun, return 1; otherwise return 0.
10. If the head of the first markable is a noun and the head of second one is a pronoun, return 1; otherwise return 0.
11. If the head of the first markable is a pronoun and the head of second one is a noun, return 1; otherwise return 0.
12. If the head of the first markable is a pronoun and the head of second one is a pronoun, return 1; otherwise return 0.
13. If the first markable is a personal pronoun, return 1; otherwise 0.
14. If the first markable is a reflexive pronoun, return 1; otherwise 0.
15. If the first markable is a possessive pronoun, return 1; otherwise 0.
16. If the first markable is a relative pronoun, return 1; otherwise 0.
17. If the first markable is a demonstrative pronoun, return 1; otherwise 0.
18. If the second markable is a personal pronoun, return 1; otherwise 0.

APPENDIX B. SEMEVAL-2010 FEATURE SETS

19. If the second markable is a reflexive pronoun, return 1; otherwise 0.
20. If the second markable is a possessive pronoun, return 1; otherwise 0.
21. If the second markable is a relative pronoun, return 1; otherwise 0.
22. If the second markable is a demonstrative pronoun, return 1; otherwise 0.
23. If the first word of the first markable is a definite article, return 1; otherwise 0.
24. If the head word of the first markable is a definite article, return 1; otherwise 0.
25. If the first word of the second markable is a definite article, return 1; otherwise 0.
26. If the head word of the second markable is a definite article, return 1; otherwise 0.
27. If one markable includes another one, return 1; otherwise return 0.
28. If one markable overlaps another one, return 1; otherwise return 0.
29. If the markables' head words agree in number and their numbers are known, then return 1; otherwise 0.
30. If the markables' head words agree in grammatical gender and their genders are known, then return 1; otherwise 0.
31. If the markables' head words agree in case (i.e. accusative, dative, nominative and genitive) and their cases are known, then return 1; otherwise 0.
32. If the markables' head words agree in pronoun person (i.e. first, second or third) and their pronoun persons are known, then return 1; otherwise 0.

Bibliography

- Frank R. Abate. *The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers*. Oxford University Press, 1996.
- Steven Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 2008.
- Ethem Alpaydin. *Introduction to Machine Learning, second edition*. MIT Press, 2010.
- Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *LREC Workshop on Linguistics Coreference '98*, pages 563–566, 1998.
- Breck Baldwin. Cogniac : A high precision pronoun resolution engine. *Jill*, 1996.
- Breck Baldwin, Tom Morton, Amit Bagga, Jason Baldrige, Raman Ch, Alexis Dimitriadis, Kieran Snyder, and Magdalena Wolska. Description of the upenn camp system as used for coreference. In *Proceedings MUC-7*, page 7, 1998.
- A.J. Bell. The co-information lattice. In *Fourth International Symp. Independent Component Analysis and Blind Signal Separation (ICA)*, pages 921–926, April 2003.
- Eric Bengtson and Dan Roth. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303, 2008.
- Shane Bergsma and Dekang Lin. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, Proceedings of the Annual Meeting of the Association for Computational

BIBLIOGRAPHY

- Linguistics (ACL), pages 33–40, Sydney, Australia, 2006. Association for Computational Linguistics.
- Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory, COLT' 98*, pages 92–100, New York, NY, USA, 1998. ACM.
- Vania Bogorny, Andrey Tietbohl Palma, Paulo Engel, and Luis Otavio Alvares. Weka-gdpm: Integrating classical data mining toolkit to geographic information systems. In *WAAMD*, pages 9–16. SBC, 2006.
- Gosse Bouma and Anne marie Mineur. Coreference related linguistic phenomena, 2006.
- Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- Andre Burkovski, Gunther Heidemann, Hamidreza Kobdani, and Hinrich Schütze. Methods for coreference visualization and annotation. Technical Report 2010/09, Intelligent Systems Group, University of Stuttgart, 2010.
- Rob Callan. *Artificial Intelligence*. Ashford Colour Press Ltd, Gosport, 2003. Department of Computer Science, Stanford University.
- Chris Callison-Burch. *Paraphrasing and Translation*. PhD thesis, University of Edinburgh, 2007.
- Claire Cardie. Using decision trees to improve case-based learning. In *ICML '93*, pages 25–32. Morgan Kaufmann, 1993.
- Claire Cardie and Kiri Wagstaff. Noun phrase coreference as clustering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 82–89, 1999.
- Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *AAAI-97*, pages 598–603. AAAI Press/MIT Press, 1997a.

BIBLIOGRAPHY

- Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18:33–44, 1997b.
- Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, 1985.
- Nancy Chinchor and Beth Sundheim. Message understanding conference (muc) 6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Linguistic Data Consortium, 2003.
- Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.
- Thomas Connolly and Carolyn E. Begg. *Database Systems: A Practical Approach to Design, Implementation and Management 2nd Ed.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.
- Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.
- D. R. Cox. *Principles of Statistical Inference*. Cambridge University Press, 2006.
- R. G. Crawford and Ian A. Macleod. A relational approach to modular information retrieval systems design. In *Proceedings of the American Society for Information Systems Annual Meeting*, volume 15, 1978.
- Aron Culotta, Michael Wick, Robert Hall, and Andrew Mccallum. First-order probabilistic models for coreference resolution. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, 2007.
- Kees Van Deemter and Rodger Kibble. On coreferring: Coreference in muc and related annotation schemes. *Computational Linguistics*, 26:629–637, 1995.
- Pascal Denis and Jason Baldridge. A ranking approach to pronoun resolution. In *Proceedings of the 20th international joint conference on Artificial intelligence*,

BIBLIOGRAPHY

- pages 1588–1593, San Francisco, CA, USA, 2007a. Morgan Kaufmann Publishers Inc. URL <http://portal.acm.org/citation.cfm?id=1625275.1625532>.
- Pascal Denis and Jason Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT '07*, pages 236–243, 2007b.
- L. R. Dice. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302, 1945.
- Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification (2nd edition)*. Wiley, New York, USA, 2001.
- Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
- Micha Elsner and Eugene Charniak. The same-head heuristic for coreference. In *Proceedings of the ACL 2010 Conference Short Papers*, Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), pages 33–37, Uppsala, Sweden, 2010. Association for Computational Linguistics.
- Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 164–172, 2009.
- Brian Everitt. *The Cambridge Dictionary of Statistics (2nd edition)*. Cambridge University Press, 2002.
- Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.
- Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.
- Jenny Rose Finkel and Christopher D. Manning. Enforcing transitivity in coreference resolution. In *HLT '08*, pages 45–48, 2008.

BIBLIOGRAPHY

- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
- Alexander Fraser and Daniel Marcu. Semi-supervised training for statistical word alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 769–776, 2006.
- Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303, 2007a.
- Alexander Fraser and Daniel Marcu. Getting the structure right for word alignment: LEAF. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 51–60, 2007b.
- Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21:1129–1164, 1991.
- Niyu Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. In *In Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170, 1998.
- Cyril Goutte, Kenji Yamada, and Eric Gaussier. Aligning words using matrix factorisation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, page 502, Morristown, NJ, USA, 2004. Association for Computational Linguistics. URL <http://dx.doi.org/10.3115/1218955.1219019>.
- D. Grossman. Using the relational model and part-of-speech tagging to implement text relevance. In *ACM CIKM*, 1992.
- David Grossman, Ophir Frieder, David O. Holmes, and David C. Roberts. Integrating structured data and text: A relational approach. *Journal of the American Society of Information Science*, 48, 1997.

BIBLIOGRAPHY

- Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 848–855, 2007.
- Aria Haghighi and Dan Klein. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1152–1161, 2009.
- Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 385–393, 2010.
- Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, Volume 11, Issue 1:10–18, 2009.
- Sven Hartrumpf. Coreference resolution with syntactico-semantic rules and corpus statistics. In *ConLL '01*, pages 1–8, 2001.
- Erhard W. Hinrichs, Sandra Kübler, and Karin Naumann. A unified representation for morphological, syntactic, semantic, and referential annotations. In *CorpusAnno '05*, pages 13–20, 2005.
- Lynette Hirschman and Nancy Chinchor. Muc-7 coreference task definition. In *Proceedings of MUC-7*. Science Applications International Corporation, 1997.
- J Hobbs. Resolving pronoun references. In *Readings in natural language processing*, pages 339–352. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986. ISBN 0-934613-11-7.
- Veronique Hoste. *Optimization Issues in Machine Learning of Coreference Resolution*. PhD thesis, Antwerp University, Antwerpen, Belgium, 2005.
- Véronique Hoste and Guy De Pauw. Knack-2002: a richly annotated corpus of dutch written text. In *LREC '06*, pages 1432–1437. ELRA, 2006.

BIBLIOGRAPHY

- Heng Ji and Dekang Lin. Gender and animacy knowledge discovery from web-scale n-grams for unsupervised person mention detection. In *Proceedings of the PACLIC 2009*, 2009.
- Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines, Methods, Theory, and Algorithms*. Kluwer/Springer, 2002.
- Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Studies in Linguistics and Philosophy. Kluwer, Dordrecht, 1993.
- Andrew Kehler, Douglas E. Appelt, Lara Taylor, and Aleksandr Simma. Competitive self-trained pronoun interpretation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 33–36, 2004.
- Manfred Klenner and Étienne Ailloud. Optimization in coreference resolution is not needed: A nearly-optimal algorithm with intensional constraints. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 442–450, 2009.
- Edgar Klenske. Textvisualisierung zur koreferenzerkennung, 2010. Studienarbeit, University of Stuttgart.
- Hamidreza Kobdani and Hinrich Schütze. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval '10*, pages 92–95, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- Hamidreza Kobdani, Alexander Fraser, and Hinrich Schütze. Word alignment by thresholded two-dimensional normalization. In *Proceedings of MT Summit XII*, 2009.
- Hamidreza Kobdani, Hinrich Schütze, Andre Burkovski, Wiltrud Kessler, and Gunther Heidemann. Relational feature engineering of natural language processing. In *Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10*, pages 1705–1708, New York, NY, USA, 2010. ACM.

BIBLIOGRAPHY

- Hamidreza Kobdani, Hinrich Schütze, Michael Schiehlen, and Hans Kamp. Bootstrapping coreference resolution using word associations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). Association for Computational Linguistics, 2011.
- Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, 2003.
- Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.
- Wessel Kraaij. *Variations on language modeling for information retrieval*. PhD thesis, University of Twente, 2004.
- Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June 2006a. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N/N06/N06-1014>.
- Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, 2006b.
- Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
- Yang Liu, Qun Liu, and Shouxun Lin. Log-linear models for word alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 459–466, 2005. URL <http://www.aclweb.org/anthology/P/P05/P05-1057>.
- Xiaoqiang Luo. On coreference resolution performance metrics. In *HLT '05*, pages 25–32, 2005.

BIBLIOGRAPHY

- Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 135–142, 2004.
- Clifford A. Lynch and Michael Stonebraker. Extended user-defined indexing with application to textual databases. In *Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings*, pages 306–317, 1988.
- D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139, 2002.
- A. McEnery, I. Tanaka, and S. Botley. Corpus annotation and reference resolution. In *ANARESOLUTION '97*, pages 67–74, 1997.
- I. Dan Melamed. A word-to-word model of translational equivalence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997.
- I. Dan Melamed. Empirical methods for MT lexicon development. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 18–30, 1998. ISBN 3-540-65259-0.
- I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.
- Alexis Mitchell, Stephanie Strassel, Mark Przybocki, JK Davis, George Doddington, Ralph Grishman, Adam Meyers, Ada Brunstein, Lisa Ferro, and Beth Sundheim. ACE-2 version 1.0. Linguistic Data Consortium, Philadelphia, 2003.

BIBLIOGRAPHY

- Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- Ruslan Mitkov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), pages 869–875, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.
- Robert C. Moore. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, 2002.
- Robert C. Moore. Improving IBM word-alignment model 1. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004.
- Robert C. Moore, Wen-Tau Yih, and Andreas Bode. Improved discriminative bilingual word alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 513–520, Sydney, Australia, 2006.
- Christoph Müller, Stefan Rapp, and Michael Strube. Applying co-training to reference resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 352–359, 2002.
- Vincent Ng. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.
- Vincent Ng. Unsupervised models for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 640–649, 2008.
- Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1396–1411, 2010.

BIBLIOGRAPHY

- Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111, 2002.
- Vincent Ng and Claire Cardie. Bootstrapping coreference classifiers with multiple machine learning algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 113–120, 2003.
- N. J. Nilsson. *Introduction to Machine Learning*. Department of Computer Science, Stanford University, 1996.
- Franz J. Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.
- Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 404–411, 2007.
- Simone Paolo Ponzetto. *Knowledge Acquisition from a Collaboratively Generated Encyclopedia*. PhD thesis, University of Stuttgart, 2010.
- Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–659, 2008.
- Sameer Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. Unrestricted coreference: Identifying entities and events in ontonotes. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC)*, Irvine, CA, September 17-19 2007a.
- Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA, June 2011a. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W11-1901>.

BIBLIOGRAPHY

- Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, Portland, Oregon, June 2011b.
- Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: A unified relational semantic representation. In *ICSC '07*, pages 517–526, 2007b.
- Long Qiu, Min-Yen Kan, and Tat-Seng Chua. A public reference implementation of the rap anaphora resolution algorithm. In *LREC '04*, 2004.
- J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501, 2010.
- Altaf Rahman and Vincent Ng. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *J. Artif. Intell. Res. (JAIR)*, 40:469–521, 2011.
- William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
- Andreas Rauber, Dieter Merkl, and Michael Dittenbach. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.
- Marta Recasens and Eduard Hovy. Blanc: Implementing the rand index for coreference evaluation. In *Journal of Natural Language Engineering*, 2010.
- Marta Recasens, Lluís Màrquez, Emili Sapena, M. Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. Semeval-2010 task 1: Coreference resolution in multiple languages. In *SemEval '10*, pages 70–75, 2010.

BIBLIOGRAPHY

- K. J. Rodriguez, F. Delogu, Y. Versley, E. Stemle, and M. Poesio. Anaphoric annotation of wikipedia and blogs in the live memories corpus, 2010.
- Lior Rokach and Oded Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Company, 2008.
- S. Russel and D. Subramanian. Provably bounded optimal agents. *Journal of AI Research*, 1995.
- Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, United Kingdom, 1994.
- Bettina Schrader. ATLAS - A new text alignment architecture. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.
- Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, pages 521–544, 2001.
- Josef Steinberger, Massimo Poesio, Mijail A. Kabadjovb, and Karel Jezek. Two uses of anaphora resolution in summarization. In *Information Processing and Management, Special issue on Summarization*, pages 1663–1680, 2007.
- Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 656–664, 2009.
- Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. Coreference resolution with reconcile. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 156–161, 2010.
- Michael Strube, Stefan Rapp, and Christoph Müller. The influence of minimum edit distance on reference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 312–319, 2002.

BIBLIOGRAPHY

- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.
- Kadim Tasdemir and Erzsébet Merényi. Exploiting data topology in visualization and clustering of self-organizing maps. *IEEE Transactions on Neural Networks*, 20(4):549–562, 2009.
- Heike Telljohann, Erhard W. Hinrichs, and Sandra Kübler. Stylebook for the tübingen treebank of written german (tüba-d/z). Technical report, University of Tübingen, 2003.
- Joel R. Tetreault. A corpus-based evaluation of centering and pronoun resolution. *Comput. Linguist.*, 27(4):507–520, 2001.
- Jörg Tiedemann. Combining clues for word alignment. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2003.
- Yoshimasa Tsuruoka. A simple c++ library for maximum entropy classification. *Tsujii laboratory, Department of Computer Science, University of Tokyo*, 2006.
- A. Ultsch and H.P. Siemon. Kohonen’s self organizing feature maps for exploratory data analysis. In *Proceedings of International Neural Networks Conference*, pages 305 – 308, 1990.
- A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.
- Yannick Versley. A constraint-based approach to noun phrase coreference resolution in german newspaper text. In *Proceedings of KONVENS 2006*. KONVENS ’06, 2006.
- Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. Coreference systems based on kernels methods. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 961–968, Stroudsburg, PA, USA, 2008a. Association for Computational Linguistics.

BIBLIOGRAPHY

- Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, and Xiaofeng Yang. Bart: A modular toolkit for coreference resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9–12, 2008b.
- Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Self-organizing map in matlab: the som toolbox. In *Proceedings of the Matlab DSP Conference*, pages 35–40, 1999.
- Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *MUC6 '95*, pages 45–52, 1995.
- N. Vlassis. A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, 2003. Retrieved 15st September 2004 from <http://www.science.uva.nl/~vlassis/cimasdai>.
- Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the International Conference on Computational Linguistics*, pages 836–841, 1996.
- Kiri Lou Wagstaff. *Intelligent clustering with instance-level constraints*. PhD thesis, Cornell University, Ithaca, NY, USA, 2002. Chair-Cardie, Claire.
- Gerhard Weiss. *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, Massachusetts, 1999.
- M. Wooldrige and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10 No 2, 1995.
- Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. Coreference resolution using competition learning approach. In *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183, 2003a.
- Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. Coreference resolution using competition learning approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 176–183, 2003b.

BIBLIOGRAPHY

Richard Zens, Evgeny Matusov, and Hermann Ney. Improved word alignment using a symmetric lexicon model. In *Proceedings of the International Conference on Computational Linguistics*, 2004.

Ying Zhang, Stephan Vogel, and Alex Waibel. Integrated phrase segmentation and alignment algorithm for statistical machine translation. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering*, 2003.