IDENTIFICATION OF ENTITY MENTIONS IN TEXT

AND THEIR COREFERENCE RESOLUTION

APPROVED BY SUPERVISORY COMMITTEE:

_____

Dr. Sanda Harabagiu, Chair

_____

Dr. Dan Moldovan

_____

Dr. Vincent Ng

To Gabriel, my parents and my friends here and there.

IDENTIFICATION OF ENTITY MENTIONS IN TEXT AND THEIR COREFERENCE RESOLUTION

by

CRISTINA NICOLAE, Eng.

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2006

PREFACE

This thesis was produced in accordance with guidelines which permit the inclusion as part of the thesis the text of an original paper or papers submitted for publication. The thesis must still conform to all other requirements explained in the "Guide for the Preparation of Master's Theses and Doctoral Dissertations at The University of Texas at Dallas." It must include a comprehensive abstract, a full introduction and literature review and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this thesis to include as chapters authentic copies of papers already published, provided these meet type size, margin and legibility requirements. In such cases, connecting texts which provide logical bridges between manuscripts are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the student's contribution to the work and acknowledging the contribution of the other author(s). The signatures of the Supervising Committee which precede all other material in the thesis attest to the accuracy of this statement.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Sanda Harabagiu, for her constant encouragement in all stages of my research, study and thesis writing. Without her valuable suggestions and guidance, this work would not have been possible. I am also grateful to Dr. Dan Moldovan and Dr. Vincent Ng for serving on my thesis committee. I owe thanks to my Engineer thesis advisor, Dr. Ştefan Trăuşan-Matu, for starting me on this journey.

I feel privileged to have had the opportunity to work at the Human Language Technology Research Institute, and I thank my friends and colleagues Finley Lăcătuşu, Cosmin-Adrian Bejan, Paul Morărescu, Rod Adams and Daniel Worlton, and the administrative staff, Cathie Ranta and Mark Hittinger, for making me feel at home in the workplace.

I thank my husband, Gabriel, for being by my side in every way. I am grateful to my parents, Gabriela and Romeo Grigore, for managing to transport their love across the ocean with the speed of thought. I am thankful to all my friends here and overseas who helped me overcome difficulties and share happiness, cheering me along, and especially to: Mihaela Onu, Silviu and Alice Nicolae, Marta Tatu, Mithun Balakrishna, Cyril Cerovic, Adrian Novischi, Ioana Lăcătuşu, Adriana Bădulescu, Ovidiu Forțu, Marian Olteanu, Iulian Nițu, Amanda Allen, Wong Wenhui, Robyn Yang, ron šult, Cécile Lorin, Belén Ortega, Julie Moore, Hekaterine, Ingrid Heyn, Andrea Santorio and, last but not least, Gijs Dekker.

November 2006

IDENTIFICATION OF ENTITY MENTIONS IN TEXT

AND THEIR COREFERENCE RESOLUTION

Publication No. _____

Cristina Nicolae, M.S.
The University of Texas at Dallas, 2006

Supervising Professor: Dr. Sanda Harabagiu

Detecting the entities in a text is a very important part of the understanding of the text. Entities represent the main concepts of the discourse, what the document "is all about". Without knowing these, the text is just a succession of words. The task of detecting entities has applications in many Natural Language Processing domains, like Machine Translation, Summarization, Information Retrieval and Question Answering— in all of which a thorough understanding of the conceptual structure of discourse is vital. This thesis proposes a method for detecting entities and their mentions in natural language text. The work is divided into two successive steps: a method for detecting all the mentions in a text, and a method for grouping these mentions together into classes that refer to the same entity. The novelties introduced in this thesis are the use of the semantic hierarchies of the WordNet lexical database to detect the entity types of nominal mentions, and a top-down, graph-based approach to clustering together mentions that refer to the same entity. It is shown that the second step benefits from the results of the first step, and that the entire system is competitive in terms of performance with the best ranked systems in the scientific community.

TABLE OF CONTENTS

LIST OF FIGURES

Number Page

LIST OF TABLES

Number Page

CHAPTER 1

INTRODUCTION

1.1 Problem Description: The Entity Detection and Tracking Task

Detecting the entities in a text is a very important part of the understanding of the text. Entities represent the main concepts of the discourse, what the document "is all about". Without knowing these, the text is just a succession of words. The task of detecting entities has applications in many Natural Language Processing domains, like Machine Translation, Summarization, Information Retrieval and Question Answering— in all of which a thorough understanding of the conceptual structure of discourse is vital.

According to the Automatic Content Extraction program ([ACE Phase 2, 2003]) EDT task guidelines:

> **An entity** is an object or set of objects in the world.
>
> **A mention** is a textual reference to an entity.

Entity detection and tracking means identifying entities and their attributes in a text. In other words, an algorithm that solves this task must find which mentions refer to which entities and group them into equivalence classes, each class corresponding to an entity. These classes have certain attributes, and the mentions themselves have attributes as well. An entity detection and tracking algorithm has to identify all this information correctly.

To exemplify the problem of entity detection and tracking, the following text has been selected from the ACE corpus.

> Paul Kariya, the superstar left wing of the Mighty Ducks of Anaheim, will not play for Canada because of a concussion he sustained Feb. 1. Kariya will be replaced by Montreal's Mark Recchi.

Figure 1. An example from the ACE corpus.

Applied on this text, an entity detection and tracking system would have to find the following information (the mention heads are underlined):

Entity 1: Person. Specific. Mentions: *Paul Kariya*, *the superstar left wing of the Mighty Ducks of Anaheim* (name), *the superstar left wing of the Mighty Ducks of Anaheim* (nominal), *he* (pronominal), *Kariya* (name).

Entity 2: Organization. Specific. Mentions: *the Mighty Ducks of Anaheim* (name).

Entity 3: Geo-political Entity. Specific. Mentions: *Anaheim* (name).

Entity 4: Organization. Specific. Mentions: *Canada* (name).

Entity 5: Geo-political Entity. Specific. Mentions: *Montreal* (name).

Entity 6: Person. Specific. Mentions: *Montreal's Mark Recchi* (name).

Let us note that **Entity 4** is an organization and not a geo-political entity, despite what a superficial look might show, because *Canada* in this context refers to a team and not the country.

The first step in the task is mention detection. This step involves finding all mentions in the text and their characteristics. Mentions are noun phrases (NPs), and can be name mentions,

nominal mentions or pronoun mentions. Mentions have the same type and genericity as the entity to which they refer. Finding the mentions also implies correctly finding their full extents and their heads. Mentions can be nested— some mentions contain other mentions, which refer to different entities, as can be seen in the following phrase:

the Mighty Ducks of Anaheim

In this case, there are two mentions which refer to two different entities: *the Mighty Ducks of Anaheim* refers to an organization and *Anaheim* to a GPE. The latter mention is nested within the former.

The second step is to identify the entities to which the text refers, by grouping the mentions using the information about them obtained in the first step. Entities are limited by the ACE specification to the five types mentioned above: PERSON, ORGANIZATION, FACILITY, LOCATION and GPE. The definition of the types, reproduced from the [ACE Phase 2, 2003] annotation guidelines, is as follows:

Person: Person entities are limited to humans. A person may be a single individual or a group.

Organization: Organization entities are limited to corporations, agencies, and other groups of people defined by an established organizational structure.

Facility: Facility entities are limited to buildings and other permanent man-made structures and real estate improvements.

Location: Location entities are limited to geographical entities such as geographical areas and landmasses, bodies of water, and geological formations.

GPE (Geo-political Entity): GPE entities are geographical regions defined by political and/or social groups. A GPE entity subsumes and does not distinguish between a nation, its region, its government, or its people.

Any entity detection and tracking system that is correct according to the ACE specifications must identify for each entity its type, its genericity class and all of the mentions in the given text that refer to the particular entity.

1.2 Mention Detection Overview

Quoting [Ittycheriah et al., 2003], mention detection is finding all mentions in the text and their type, level and genericity. Finding these mention characteristics is the first step in an entity detection and tracking system. The mention attributes extracted in this phase serve in the second (tracking) phase as selectional constraints or features. The most work in this area was done for detecting the type of named entities, but a few recent approaches take into consideration other types of mentions and other attributes.

The first approach discussed is by [Ittycheriah et al., 2003], who proposed a system for identifying and tracking entity mentions in a document, as part of the Automatic Content Extraction evaluation for entity detection and tracking. The researchers divided the task into two subproblems, mention detection and mention tracking, and adopted different methods for each. For mention detection, they employed two complementary named entity recognizers and combined them with other syntactic and lexical sources of information into a maximum entropy probabilistic framework. The result was a high performance. For mention tracking, they proposed a new statistical approach, which gave relevancy scores to pairs of mentions

and then clustered them with a greedy algorithm that operated in the order of apparition. The model used binary lexical, syntactic and semantic features. The performance of the system in the evaluation was satisfactory.

A second approach was authored by [Florian et al, 2003]. It consists of a statistical model for multilingual entity detection and tracking— the framework was not designed to be specific to a certain language, but instead worked for three different languages: Arabic, Chinese and English. The researchers also separated the task into two subtasks, the mention detection and the entity tracking. The mention detection is recast as a classification problem, with a linear and a log-linear classifier and lexical, syntactic and semantic features. In addition, the authors combined multiple pre-existing mention taggers. A few of the features were language-independent (e.g. the part of speech, dictionary information), while the others were language-specific, corresponding to each of the three languages considered. Entity tracking used a maximum entropy model with lexical and syntactic features, and a novel probabilistic coreference decoding algorithm. The experiments proved that the systems performed remarkably well, for all three languages.

1.3 Coreference Resolution Overview

As defined by [Ng, 2005], coreference resolution (also known as anaphora resolution or entity tracking) is the problem of determining which noun phrases in a text or dialogue refer to which real-world entity. Coreference resolution is an important preliminary subtask in discourse processing tasks like Question Answering, Segmentation, Summarization and Information Extraction. Knowing which noun phrases refer to the same entity is a useful piece of information when searching for a solution in the previously mentioned complex

tasks; without this information, an algorithm has a performance penalty from the start. As written by [Soon et al., 2001], in particular, information extraction (IE) systems like those built in the DARPA Message Understanding Conferences ([Chinchor, 1998], [Sundheim, 1995]) have revealed that coreference resolution is such a critical component of IE systems that a separate coreference subtask has been defined and evaluated since MUC-6 ([MUC-6, 1995]). Coreference resolution basically means linking all mentions that refer to an entity (anaphors) to mentions previously appeared in the text that are part of the same entity (antecedents).

1.3.1 Early Studies

The complex problem of coreference resolution was tackled in the beginning by considering only a subpart: pronominal anaphora resolution. [Hobbs, 1986] proposed both a naïve syntactic algorithm based on sentence parse trees and selectional constraints and a semantic algorithm based on knowledge in the form of predicate calculus axioms (which made up a lexicon). This knowledge was augmented by applying semantic inference rules on the axioms to obtain new axioms. Hobbs did not implement his methods, but tested them by hand on a small set of sentences.

[Ge et al., 1998] introduced a statistical approach to pronominal anaphora resolution. This approach was different from earlier work by not relying on knowledge, instead using a very small corpus from Penn Wall Street Journal Tree-bank text ([Marcus et al., 1993]). They incorporated multiple anaphora resolution factors into a statistical framework: the distance between the pronoun and the proposed antecedent, the gender/number/animacity of the proposed antecedent, governing head information and noun phrase repetition. The

gender/number/animacity information is learned through their method of unsupervised learning.

[Mitkov, 1998] also rejected a knowledge-based system, one that would rely heavily on linguistic and domain knowledge, and instead presented a robust, knowledge-poor approach to resolving pronouns in technical manuals. The input was checked for agreement and for a number of antecedent indicators. All candidates were assigned scores by each indicator and the candidate with the highest score was returned as the antecedent. The method outperformed all previous efforts.

1.3.2 Philosophy of Reference

In his two papers from 1979 and 2002, Jerry Hobbs discussed theoretical aspects of coreference resolution. [Hobbs, 1979] treated the problem of coherence and coreference. He started by presenting the three requirements for a theory of coherence: first, that we should be able to explain the function of each of the coherence relations; second, that the cohesive relations studied by [Halliday and Hasan, 1976] (identity, similarity and subpart) could be seen as deriving from the coherence relations; and finally, that the relations must be computable. He continued with the inference component and its four aspects: data, representation, operations and control. He then proceeded to define some coherence relations: elaboration, parallel and contrast. Finally, he talked about resolving coreference based on coherence information: resolving reference against prior discourse, resolving reference against a world model and resolving reference against an alternate representation.

[Hobbs, 2002] discussed the definite determiner “the” and its uses to express the relation between the entity referred to by the noun phrase and the description provided by the noun phrase. According to Jerry Hobbs, who examined several hundred examples of the use of “the” in a diverse corpus, the examples can be classified into six categories: mutually known entities, directly anaphoric “the”, indirectly anaphoric “the”, bridging, determinative definite noun phrases and generic definite noun phrases.

1.3.3 Statistical Techniques

Research in coreference resolution started by leaning on heavy knowledge, but recent data-driven approaches compare in performance with the knowledge-based ones. After resolving the pronominal anaphora, researchers turned to other kinds of anaphora: names and common nouns. The supervised machine learning approaches proposed by [Soon et al., 2001], [Ng and Cardie, 2002], [Yang et al., 2003] and [Luo et al., 2004] recast coreference resolution as a binary classification task. The classification target functions for a noun phrase (NP) in the text are “coreferent” and “non-coreferent”. The classification is done by a machine learning algorithm, which in the first phase trains on the development set of an annotated corpus and in the second phase is applied to the input text to classify all its NPs. After the classification has ended, the coreference algorithm applies a clustering method to obtain equivalence classes from the NPs in the text, according to their pairwise coreference confidence. Each of these equivalence classes corresponds to a real-world entity. The output of the algorithm is the input text annotated with the obtained coreference information.

Other data-driven approaches are weakly supervised ([Muller et al., 2002], [Ng and Cardie, 2003a], [Ng and Cardie, 2003b]) and unsupervised ([Bean and Riloff, 2004]). While a few of

the approaches rely on a simple machine learning algorithm, some of them do a selection based on the competition between algorithms ([Muller et al., 2002], [Ng and Cardie, 2003a], [Ng, 2005]). Some go even further, to a competition between candidates ([Yang et al., 2003]) or a selection among all possible methods of optimization ([Ng, 2004], [Ng, 2005]).

1.4 The Importance of Entity Detection and Tracking

The entity detection and tracking task obtains information about the entities and about the textual mentions that refer to them. For each word of the text that is relevant to the text because it refers to a real-world entity, the resolver returns the following information: what *type of entity* it is (PERSON, ORGANIZATION, LOCATION, FACILITY, GPE), what *type of mention* it is (name, nominal, pronominal) and how *generic* it is (generic or specific). This information helps in the text's understanding, which permits a deeper level of analysis to the applications that use it. Entity detection and tracking is actually a subpart of information extraction ([Cunningham, ?]), combining the entity detection with the coreference resolution components into a more complex task than the simple knowledge of coreference of the MUC task ([MUC-6, 1995]). This rich knowledge is very helpful in applications like Machine Translation, Summarization, Information Retrieval and Question Answering.

1.4.1 EDT and Machine Translation

Machine Translation (automatic text translation from one language to another) is a difficult task because behind this simple procedure there lies a complex cognitive operation. For example, to decode the meaning of the source text in its entirety, the translator must interpret and deeply analyze all the features of the text, of which an important part are the entities.

Knowing the type of the entity, for example, serves as a disambiguation when the translator is confronted with a word that can be translated differently depending on the context.

> **English:** Kariya will be replaced by Montreal's Mark Recchi.
>
> **Babelfish English to French automatic translation:** Kariya sera remplacé par la marque Recchi de Montréal.
>
> **Correct French translation:** Kariya sera remplacé par Mark Recchi de Montréal.

As can be seen from the previous example, Altavista's Babelfish automatic translator did not realize that *Mark* is the name of a person, and translated it into French by considering it a common noun (*the mark*). If the translator had the information that the word was a person, it would know to reproduce it at translation. To note is that capitalization is not a perfect indicator of a name; there are many texts in which proper names are not capitalized.

1.4.2 EDT and Summarization

Summarization is the automated process of obtaining a summary, or an abstract, from a document. Knowing the entities can improve the task, by increasing the text understanding. Let us consider an example.

> Mrs. Peacock killed Mr. Body in the conservatory.
> She killed him with the revolver.

In this situation, the summary would be:

Mrs. Peacock killed Mr. Body in the conservatory with the revolver.

To obtain such a summary, a program must know that *she* is in fact *Mrs. Peacock* and that *him* refers to *Mr. Body*. This information is a result of the EDT task. Without the information, the summary would not incorporate all the facts in the text, and would be incorrect.

1.4.3 EDT and Information Retrieval

Information Retrieval searches, organizes and analyzes the data in a collection of documents based on user queries. The most popular such collection of documents is the World Wide Web. Retrieval is not only at a lexical level— it can be semantic as well. That is, if the user query is *I.B.M.*, the results returned should also contain documents referring to *International Business Machines Corporation*, considering these are the same entity. This is where the EDT task is needed. By having information about the entities, a search engine can improve the relevance of the results it retrieves.

1.4.4 EDT and Question Answering

The same applies to Question Answering, which is the automatic answering of questions based on similarity with a collection of documents. Since the answering is done by comparing the text of the question with the text in the collection, it is imperative to know which of the words refer to the same things, because the phrasing of the question and the collection can differ, and instead of generating all the possible questions, it is easier to use the coreference information.

Text fragment: President Bush was born on July 6, 1946, in New Haven, Connecticut, and grew up in Midland and Houston, Texas.

Question: When did the 43rd American president arrive into this world?

The answer is hard even for people, but becomes harder for the computer because the question does not refer to the name of the president, while the text contains the name. The correct answer:

on July 6, 1946

can be inferred by knowing that *President Bush* and *the 43rd American president* are the same real-world entity.

1.5 Thesis Goal

The thesis goal is to present a pipelined approach to entity detection and tracking, by, firstly, detecting all mentions in a text and, secondly, grouping together these mentions into classes that correspond to the same real world entity. The thesis will propose methods to solve both of these steps, and will detail their characteristics, algorithms and experimental results.

1.6 Thesis Layout

This chapter introduces the problem of entity detection and tracking, discusses mention detection and entity tracking (or coreference resolution) by presenting some of the previous work done in the respective fields, and exemplifies the importance of entity detection and tracking in natural language documents.

Chapter 2 presents the types of linguistic resources employed in this work, comprising the WordNet electronic dictionary, three named entity recognition systems and a word sense disambiguation system. Despite of the fact that the named entity recognizer used here is developed in-house, the discussion of the three systems offers an insight in the workings of such a program. The three resources have all proven relevant in solving the entity detection and tracking task.

Chapter 3 presents the methodology for mention detection. It starts with an overview of the problem and continues with describing the steps of the method. Mention detection is cast as a

classification problem, and the features on which the classification algorithm relies are detailed.

Chapter 4 describes the graph algorithm for entity tracking called BESTCUT. The problem of entity tracking is introduced and the entity tracking algorithm is discussed with an emphasis on the classification features and the clusterization method, with all the steps it employs. This chapter also presents the advantages and disadvantages of BESTCUT.

Chapter 5 presents the experiments and results performed on both the mention detection subsystem and the coreference resolution subsystem, separately and combined. It is empirically proven that the best performance of the complete system is achieved when the two subsystems are combined. The results are competitive with the best results published to-date in the coreference resolution problem.

The thesis concludes with a recapitulation of the main points presented in the document.

CHAPTER 2

LINGUISTIC RESOURCES

2.1 The WordNet Electronic Dictionary

According to [Miller, 1995], WordNet is an online lexical database designed for use under program control. It is encoded in computer-readable form, and combines the features of a dictionary and a thesaurus. This makes it a valuable resource in natural language understanding, by giving the computer information that until now was only available to humans.

The lexical database contains English nouns, verbs, adjectives, and adverbs (open-class words). The words are grouped into sets of synonyms (synsets), each representing a lexicalized concept. The synsets are linked through semantic relations. Each word can be a noun, a verb, an adjective or an adverb, or combinations of these, depending on its context.

An example of an entry in WordNet for the word *"ferret"* is illustrated in Figure 2.

Noun

1. (n) black-footed ferret, ferret, Mustela nigripes (musteline mammal of prairie regions of United States; nearly extinct)

2. (n) ferret (domesticated albino variety of the European polecat bred for hunting rats and rabbits)

Verb

1. (v) ferret (hound or harry relentlessly)

2. (v) ferret (hunt with ferrets)

3. (v) ferret out, ferret (search and discover through persistent investigation) "She ferreted out the truth"

Figure 2. The WordNet entry for the word *"ferret"*.

The noun *"ferret"* has two senses, which means it is a part of two different synsets. Each sense is associated with the words of the synset corresponding to it and with the definition of the word (the gloss). The verb *"ferret"* has three senses, and is part of three synsets. Sometimes the WordNet entries also contain examples, like *"She ferreted out the truth"*, which help the user to better understand some of the senses. A monosemous word is a word that has only one sense. A polysemous word is a word that has more than one sense, and the senses are interrelated. For example, a polysemous word is *"school"*, which can mean both a building and an institution, because these two senses are connected. The two senses are called the polysemous senses of the word *"school"*.

Statistical information presented in [Miller, 1995] said that WordNet contained more than 118,000 different word forms and more than 90,000 different word senses, or more than 166,000 word-sense pairs. Approximately 17% of the words in WordNet were polysemous;

approximately 40% had one or more synonyms. Up-to-date statistics on the WordNet main website[1], presented in Table 1 and Table 2**,** show the evolution of WordNet.

Table 1. WordNet statistics in 2006

POS	***Unique Strings***	***Synsets***	***Total Word-Sense Pairs***
Noun	117097	81426	145104
Verb	11488	13650	24890
Adjective	22141	18877	31302
Adverb	4601	3644	5720
Totals	**155327**	**117597**	**207016**

Table 2. WordNet monosemy and polysemy statistics in 2006

POS	***Monosemous Words and Senses***	***Polysemous Words***	***Polysemous Senses***
Noun	101321	15776	43783
Verb	6261	5227	18629
Adjective	16889	5252	14413
Adverb	3850	751	1870
Totals	**128321**	**27006**	**78695**

Table 3. Semantic relations in WordNet

Semantic Relation	***Syntactic Category***	***Examples***
Synonymy (symmetric)	N, V, Adj, Adv	N: student/pupil V: drink/imbibe Adj: red/crimson Adv: slowly/tardily
Antonymy (symmetric)	N, V, Adj, Adv	N: night/day V: run/idle Adj: white/black Adv: violently/nonviolently
Hyponymy (transitive)	N	bus/car
Meronymy	N	air bag/car
Troponymy	V	limp/walk
Entailment	V	snore/sleep

[1] http://wordnet.princeton.edu/man/wnstats.7WN

WordNet has six semantic relations that can occur between synsets. Table 3 illustrates these relations.

Synonymy is the relation of similarity between two words.

Antonymy is the relation of oppositeness of two words.

Hyponymy is the relation between a subordinate and a superior. (Reverse relation is **hypernymy**.)

Meronymy is the part-to-whole relation. (Reverse is **holonymy**.)

Troponymy is a relation between a manner of doing an action and the action (has corresponding relation **hyponymy** for nouns).

Entailment is an implication between two actions.

2.2 Named Entity Recognition Systems

Named Entity Recognition (NER) means identifying named entities in text and their types. NER is an important step in mention detection, because named entities play a very important role in the significance of a discourse. Name, nominal and pronominal mentions cannot be treated as the same type of words; each needs a different, specific method of resolution. Each resolution has its level of performance, and, typically, named entity recognition is the closest in performance to human performance.

Because named entity recognition is not in the scope of this thesis, an in-house named entity recognizer was used, which is similar to Nymble, the first system that will be presented in this section. It was used to identify five named entity types: PERSON, ORGANIZATION, LOCATION, FACILITY and GPE, which are the same types of entities that the mention detection algorithm presented in this thesis recognizes.

2.2.1 [Bikel et al., 1997]'s Nymble NER System

Nymble was introduced in [Bikel et al., 1997] as a statistical, hidden Markov model-based approach to finding names and other non-recursive entities in text, as per the MUC-6 task. The named entity recognizer used a slightly-modified version of the hidden Markov models used in speech recognition. The approach considered the raw text to annotate with named entity information as though it had been initially marked with the correct named entities and then had passed a noisy channel, similarly to a speech processing system; and the task was to recreate the original named entity annotations, thus solving the named entity recognition.

The model was a hidden Markov model with eight internal states, corresponding to the name classes, and the start and end of sentence states. The generation of the words and the name classes went as follows: first, select a name class, conditioned by the previous name class and the previous word; then, generate the first word of that name class, conditioned by the current and previous name classes; finally, generate all subsequent words of that name class, each of them being conditioned on its immediate predecessor. They used smoothing and back-off models to make up for the sparseness of data. The three steps were repeated until the entire observed word sequence was generated, and then they used the Viterbi algorithm to search the space of assignments, maximizing the probability of the name classes given the words.

The Nymble recognizer performed at near-human performance, with results that surpassed 90% accuracy. The system identified the named entity types in the MUC task, namely organizations, persons, locations, times, dates, percentages and money amounts.

2.2.2 [Carreras et al., 2002]'s NER System

In the 2002 CoNLL named entity extraction task, [Carreras et al., 2002] obtained the best performance both for Spanish and Dutch. Their system broke the task of named entity extraction into named entity recognition (NER) and named entity classification (NEC). The two subtasks were tackled independently, with two different machine learning-based modules, making use of binary AdaBoost classifiers.

The features were window-based, which means they took into account a window rooted in a word, on whose words they applied the features. Some of the features used were: word form, part-of-speech, orthographic features, word type patterns, bag-of-words, trigger words, gazetteer features and left predictions.

The NER module combined three local classifiers: BIO, Open-Close&I and Global Open-Close. The first classifier, BIO, tagged each word for beginning an NE (B), being inside an NE (I) and being outside an NE (O). The second one, Open-Close&I, detected the word that opened an NE and the word that closed it. In addition, each word inside the current NE was verified against the I module of the BIO classifier. The Global Open-Close classifier searched for opens and closes of NEs, but the set of NEs was produced with global inference.

The NEC module assigned a type to each detected named entity; the classification was done independently, which means no NE type was considered to depend on the types of previous NEs. This is the most important difference between this method and Nymble—in Nymble, all

decisions were taken in a bigram model (every NE label depended on the previous NE labels assigned).

The BIO classifier performed the best of all in the NER task, while the overall NEE task gave lower results than the NER module. Putting the two submodules one after another propagated errors and degraded the performance for the main task. The program only classified the NEs into locations, organizations and persons, putting the rest in the misc. category. The system performed better in Spanish than in Dutch, but it was designed to be language-independent.

2.2.3 [Florian et al., 2003]'s NER System

In the same conference, CoNLL, but in the next year, in the English and German task, [Florian et al., 2003] proposed a new named entity recognition system through classifier combination. These classifiers were more complex than the previous system's— there were four different classifiers: a robust linear classifier, a maximum entropy classifier, a transformation-based learning classifier and a hidden Markov model classifier (similar to the one proposed in [Bikel et al., 1999], presented at the beginning of this section).

All of the algorithms labeled each word with a tag corresponding to its position relative to a named entity: at the start or end of an entity, inside or outside of an entity. They used a diverse set of features, some of which were: words, lemmas, POS tags, text chunks, prefixes and suffixes of words in a window surrounding the current word, a word feature flag, gazetteer information and the output of two other NE classifiers.

The classifiers were all run on the data, and their statistical results were combined through various weighting fashions: the equal voting, in which all classifiers have the same weight in the final result, and the weighted voting, in which the weights are based on the probabilities of error of the classifiers. As an alternate combination method, they used, instead of voting entirely for one class, giving partial credit to alternate classifications. Finally, they used the robust risk minimization classifier to compute the function of combination of the classifiers. Incorporating the gazetteers and the output of the other two named entity systems increased the performance. The use of robust risk minimization proved to obtain the best performance out of all methods of combination.

2.3 Word Sense Disambiguation System

Word Sense Disambiguation means identifying in a text the correct sense of a word that has multiple senses, depending on the context in which it appears. The disambiguation of senses has proved very important in many applications, and in particular in Entity Detection and Tracking. Intuitively, one can see how words with multiple senses can induce confusion and incorrectness into a language understanding algorithm. For instance, if we have the following two sentences:

> The man went into the bank to take out some money.
> A day before, his boy had played on the bank of the river and fallen into the water.

and we are asked to find the entities in this short text, a program that does not know about multiple word senses and only takes lexical aspect into consideration would say the first *"bank"* and the second *"bank"* refer to the same entity, which is completely incorrect. This simple example is only one of the many such occurrences where a word sense disambiguator is needed.

For the purpose of this research, the word sense disambiguator developed by [Mihalcea and Csomai, 2005], SenseLearner[2], was used. This is a minimally supervised system that tags all words in the text input with their WordNet sense numbers. The algorithm was intended to be general enough to not depend on the document domain and efficient enough to be able to disambiguate large collections of text. The algorithm takes a preprocessed input, annotated with part of speech tags and word lemmas, and learns semantic models for predefined categories of words like nouns or verbs. Additional models can be programmed by the user. The learning is done using the Timbl memory based learning algorithm [Daelemans et al., 2001]. During training, the program obtains a feature vector for each word, and the target function is the word paired with its WordNet sense. In the testing stage, for the test examples the program associates each word with a feature vector and classifies it, by predicting its target word-sense pair. If the word in the outcome, obtained through classification, is identical to the word to be classified, then the sense is attached to the word to be classified; otherwise, the word will be annotated at a later time.

Here is an example of the running of this algorithm on a part-of-speech and lemma tagged text.

[2] SenseLearner is publicly available for download at http://lit.csci.unt.edu/~senselearner.

Initial text

Dangue/dangue/NNP Rewaka/rewaka/NNP also/also/RB said/said/VBD the/the/DT council/council/NN urged/urged/VBD protection/protection/NN of/of/IN international/international/JJ aid/aid/NN workers/worker/NNS and/and/CC steps/step/NNS to/to/TO return/return/VB the/the/DT country/country/NN to/to/TO stability/stability/NN ././.

Text after running SenseLearner

Dangue/dangue/NNP Rewaka/rewaka/NNP also/also/RB said/said/VBD the/the/DT council/council/NN#1 urged/urged/VBD protection/protection/NN#1 of/of/IN international/international/JJ#1 aid/aid/NN#3 workers/worker/NNS#1 and/and/CC steps/step/NNS#1 to/to/TO return/return/VB#2 the/the/DT country/country/NN#2 to/to/TO stability/stability/NN#1 ././.

We can observe in this result that most of the word senses were identified correctly, but there are a few incorrect pairings as well. Some of the correct ones, with their glosses taken from WordNet:

c*ouncil:1* means a body serving in an administrative capacity;

protection:1 means the activity of protecting someone or something;

aid:3 is a gift of money to support a worthy person or cause;

step:1 is any maneuver made as part of progress toward a goal;

state:2 is a politically organized body of people under a single government.

The incorrect ones are *return:2* and *stability:1*. *Return:2* means to give back, while *stability:1* is the quality or attribute of being firm and steadfast. The senses that should have been chosen instead are *return:3* (go back to a previous state) and *stability:2* (a stable order).

CHAPTER 3

A METHODOLOGY FOR MENTION DETECTION

3.1 The Problem of Mention Detection

Understanding a document written in natural language is a very important first step in natural language processing applications. Because of the large amount of data available, developing automatic processing methods is required. In order to process a document, an information extractor must first identify its key points: the entities. According to the definition introduced by [ACE Phase 2, 2003], an **entity** is an object or set of objects in the world. Entities are referred to in the text by noun phrases called mentions. A **mention** is a textual reference to an entity. Identifying the entities is done by a two-step procedure:

Step 1. Mentions of each entity are detected;

Step 2. Mentions corresponding to the same entity are grouped together.

The first step is also known as *the mention detection step*, while the second is also known as *the entity tracking step*. Mention detection is performed by first identifying the mentions' boundaries, i.e. the position in text where the mentions start and end, and their heads, i.e. the main words that represent the mentions. After this, the mentions are classified according to (a) their entity type or semantic class (e.g. PERSON, ORGANIZATION, LOCATION, FACILITY, or GPE); (b) their mention type (e.g. name, nominal or pronominal) or (c) their genericity (e.g. general or specific).

The method described in this section attempts to solve the first step of automatically detecting entities and their mentions in texts, the mention detection. The method focuses on obtaining the semantic class of nominal mentions, which is used further in Step 2. Chapter 4 of this thesis will tackle this second step, i.e. the problem of grouping together the mentions referring to the same entities. The novelty of the mention detection method presented here consists in using semantic information from the WordNet lexical database ([Fellbaum, 1998]) to detect the semantic class of the mentions and using information about word senses.

To illustrate the problem of nominal mention detection, a text has been selected from the MUC-6 corpus ([MUC-6, 1995]), and it is shown in Figure 3. The figure emphasizes the heads of the mentions that are a part of one of the five semantic classes exemplified above: PERSON, ORGANIZATION, LOCATION, FACILITY, or GPE.

A **spokesman** for the **company** said American **officials** "felt talks had reached a point where mediation would be helpful." Negotiations with the **pilots** have been going on for 11 months; talks with flight **attendants** began six months ago.	PERSON: spokesman, officials, pilots, attendants ORGANIZATION: company

Figure 3. A text example from MUC-6 and the nominal mentions it contains, with their semantic classes or entity types.

3.2 The Methodology for Mention Detection

Because the entity tracking algorithm presented in Chapter 4 relies heavily in its initial step on knowing entity types, a method was developed for recognizing entity types for nominal mentions. This statistical approach uses the ACE corpus, which is annotated with mention and entity information, as data in a supervised machine learning algorithm.

Six entity types were assigned: PERSON, ORGANIZATION, LOCATION, FACILITY, GPE and UNK (for those who are in neither of the former categories), and two genericity outcomes:

GENERIC and SPECIFIC. Only the intended values of the mentions from the corpus were considered. This was motivated by the fact that entity tracking links together mentions according to the context in which they appear, and not in a general way. Experiments discovered that the use of word sense disambiguation improves the performance significantly (a boost in score of 10%), therefore information about word senses was obtained from the word sense disambiguation program taken from [Mihalcea and Csomai, 2005] and described in Section 2.3.

The method was reported in the Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006) as [Nicolae and Nicolae, 2006].

3.2.1 Obtaining Mention Boundaries

In order to identify the extents and heads of the mentions in the text, a five-step procedure was used. It is illustrated in Figure 4. Given a plain text document, the first step applies a heuristic sentence splitter on it to partition it into sentences. These sentences are tokenized to obtain all the word tokens; this is also done heuristically, by taking as separators white space and punctuation. This information is used as input for the Brill part of speech tagger, which labels each word with its part of speech (e.g. NN, NNP, PRP$). With the text tokenized and tagged, Collins' parser builds its parse tree structure, which is used to select the nominal, pronominal and name mention extents and heads. The main constraint in selecting the mention extents and heads is that no two mentions can have the same head; and if two such mentions are found, the one with the largest extent is preferred.

Detecting Mention Boundaries

Input: A plain text document.
Output: Its mentions' boundaries and heads.

Step 1. Sentence splitting (heuristically).
Step 2. Tokenization.
Step 3. Part of speech tagging with Brill's part of speech tagger.
Step 4. Parse tree generation with Collins' parser.
Step 5. Selection of nominal, pronominal and name mentions from the text.

Figure 4. Detecting mention boundaries.

After the mentions are selected, an in-house named entity recognizer is applied on them to identify the types of the named entities, and SenseLearner, the word sense disambiguator presented in Section 2.3, is applied to obtain the WordNet senses of the mentions.

3.2.2 Mention Detection as Classification

The choice of the semantic class for each mention in a text is cast as a classification problem. The classification algorithm employed is a maximum entropy statistical model introduced by [Berger et al., 1996].

Given a linguistic phenomenon that needs to be modeled statistically (e.g. detection of mentions), a set of models can be derived, but the maximum entropy philosophy dictates that we select the model with the most uniform distribution. The modeling predicts an outcome *y* based on statistical data gathered about the phenomenon. In generating *y*, the process may be influenced by some contextual information *x*. Feature functions are used to express the statistics of the sample; they associate the sets of attributes *x* with the outcomes *y*.

To measure the uniformity of the conditional probability that the model assigns to outcome *y* in context *x*, $p(y \mid x)$, the mathematical definition of conditional entropy can be used:

$$H(p) = -\sum_{x,y} \tilde{p}(x)\, p(y \mid x) \log p(y \mid x)$$

Equation 1: Mathematical definition of conditional entropy.

In Equation 1, $\tilde{p}(x)$ is the empirical distribution of *x* in the training sample. The conditional entropy $H(p)$ is used for defining [Berger et al., 1996]'s Maximum Entropy Principle, illustrated in Figure 5. The Principle postulates that entropy must be maximized in Equation 1 such that the selected model has the most uniform distribution, which conforms to the maximum entropy philosophy.

Maximum Entropy Principle

To select a model from a set C of allowed probability distributions, choose the model $p^* \in C$ with maximum entropy $H(p)$:

$$p^* = \arg\max_{p \in C} H(p)$$

Figure 5. Maximum Entropy Principle by [Berger et al., 1996].

The maximum entropy principle presents us with a problem in constrained optimization: find the $p^* \in C$ that maximizes $H(p)$. To address this problem, the method of Lagrange multipliers is applied. The Lagrange multipliers λ_i are weighing parameters— each multiplier associates a weight to a feature function. To solve the maximum entropy problem, the Lagrange multipliers are obtained from a maximum likelihood formalization. [Berger et al., 1996] have shown that the maximum entropy problem is a dual of the maximum likelihood problem. Maximum likelihood aims to find the multipliers that maximize a function $\Psi(\lambda)$ defined as

$$\Psi(\lambda) \equiv \Lambda(p, \lambda),$$

where $\Lambda(p, \lambda)$ is a Lagrangian defined as: $\Lambda(p, \lambda) \equiv H(p) + \sum_i \lambda_i (p(f_i) - \tilde{p}(f_i)$, in which f_i are feature functions, $p(f_i)$ is the expected value of f_i with respect to the model $p(y \mid x)$, and $\tilde{p}(f_i)$ is the expected value of f_i with respect to the empirical distribution $\tilde{p}(x, y)$. The Lagrange multipliers are found through optimization numerical methods. The optimization method specifically constructed for the maximum entropy problem is the Improved Iterative Scaling algorithm, which applies for non-negative feature functions f_i. The optimization method is illustrated in Figure 6.

Improved Iterative Scaling

Input: Feature functions $f_1, f_2, \ldots f_n$; empirical distribution $\tilde{p}(x, y)$
Output: Optimal parameter values λ^*_i; optimal model p_{λ^*}

Step 1. Start with $\lambda_i = 0$ for all $i \in \{1, 2, \ldots, n\}$
Step 2. Do for each $i \in \{1, 2, \ldots, n\}$:

a. Let $\Delta\lambda_i$ be the solution to

$$\sum_{x,y} \tilde{p}(x) p(y \mid x) f_i(x, y) e^{\Delta\lambda_i f^{\#}(x,y)} = \tilde{p}(f_i),$$

where $f^{\#}(x, y) \equiv \sum_{i=1}^{n} f_i(x, y)$.

b. Update the value of λ_i according to: $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$

Step 3. Go to Step 2 if not all the λ_i have converged

Figure 6. Improved Iterative Scaling used to obtain the Lagrange multipliers.

Until all multipliers have converged, the Improved Iterative Scaling algorithm updates them with the increments calculated in step 2a of the algorithm in Figure 6.

In the experiments for mention detection, the *maxent*[3] toolkit was used. For creating training instances, an outcome was associated with each markable (i.e. noun phrase, or NP) detected in the training files: the markables that were present in the key files[4] took their outcome from the key file annotation, while all the other markables were associated with outcome UNK. Then, a training example was created for each of the markables, with the feature vector described in Section 3.2.3 and as target function the outcome. The outcome can be of three different types: (1) the entity type (one member of the set PERSON, ORGANIZATION, LOCATION, FACILITY, GPE and UNK); (2) the genericity information (GENERIC or SPECIFIC); and (3) a combination between the two (pairwise combinations of the entity types set and the genericity set, e.g. PERSON_SPECIFIC). The training data provided by ACE[5] was used; the features are detailed in Section 3.2.3.

3.2.3 Feature Representation for Mention Detection

The feature set consists of (a) WordNet features, (b) lexical features, (c) syntactic features and (d) intelligent context features, displayed in Table 4 and explained in the current section. A *WordNet-equivalent-concept* for an entity type is a word-sense pair from WordNet whose gloss covers a part of the [ACE Phase 2, 2003] specification of that entity type. The *WordNet-equivalent-concepts* were extracted by hand by studying the [ACE Phase 2, 2003] specifications. Figure 7 enumerates a few *WordNet-equivalent-concepts* for entity class PERSON (e.g. CHARACTER#1), with their hierarchy of hyponyms (e.g. Frankenstein#2).

---

[3] This toolkit is available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

[4] The key files are the training files of the corpora annotated by humans with coreference information.

[5] The ACE-2 corpus is available at http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T11.

Table 4. The features for mention detection.

#	*Category*	*Feature name*	*Feature description*
1	WordNet	*is-a-TYPE*	true if the mention is of entity type TYPE; five features
2		*WN-eq-concept-hyp*	true if the mention is in hyponym set of WordNet equivalent concept; 41 features
3		*WN-eq-concept-syn*	true if the mention is in synonym set of WordNet equivalent concept; 41 features
4	Lexical	*stem-sense*	pair between the stem of the word and the WordNet sense given by the word sense disambiguator
5	Syntactic	*pos*	part of speech of the word given by the part of speech tagger
6		*is-modifier*	true if the mention is a modifier in another noun phrase
7		*modifier-to-TYPE*	true if the mention is a modifier to a TYPE mention
8		*in-apposition-with*	TYPE of the mention our mention is in apposition with
9	Intelligent context	*all-modifiers*	the nominal, adjectival and pronominal modifiers in the mention's parse tree
10		*preps*	the prepositions right before and after the mention's parse tree

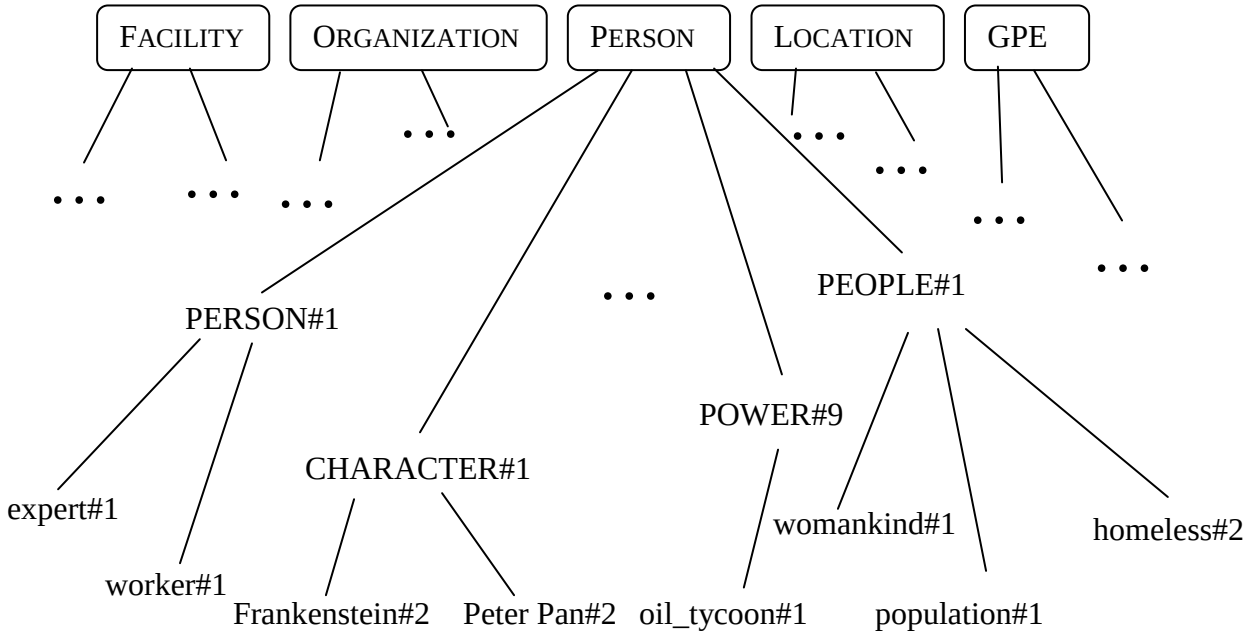


Figure 7. Part of the hierarchy containing 41 *WordNet-equivalent-concepts* for the five entity types, with all their synonyms and hyponyms. The hierarchy has 31,512 word-sense pairs in total.

Table 5 reproduces the WordNet glosses for each *WordNet-equivalent-concept* for the five entity types.

Table 5. The *WordNet-equivalent-concepts* extracted by hand for the five entity types.

PERSON	**PERSON: 1.** (7229) person, individual, someone, somebody, mortal, human, soul -- (a human being; "there was too much for one person to do") **CHARACTER: 1** (16) fictional character, fictitious character, character -- (an imaginary person represented in a work of fiction (play or film or story); "she is the main character in the novel") **CHARACTER: 4.** (9) character, role, theatrical role, part, persona -- (an actor's portrayal of someone in a play; "she played the part of Desdemona") **PEOPLE: 1.** (559) people -- ((plural) any group of human beings (men or women or children) collectively; "old people"; "there were at least 200 people in the audience") **PEOPLE: 2.** (94) citizenry, people -- (the body of citizens of a state or country; "the Spanish people") **PEOPLE: 3.** (40) multitude, masses, mass, hoi polloi, people -- (the common people generally; "separate the warriors from the mass"; "power to the people") **PEOPLE: 4.** (5) people -- (members of a family line; "his people have been farmers for generations"; "are your people still alive?") **POWER: 9.** (3) baron, big businessman, business leader, king, magnate, mogul, power, top executive, tycoon -- (a very wealthy or powerful businessman; "an oil baron") **WORLD: 1.** (335) world, human race, humanity, humankind, human beings, humans, mankind, man -- (all of the inhabitants of the earth; "all the world loves a lover"; "she always used `humankind' because `mankind' seemed to slight the women")
ORGANIZATION	**ORGANIZATION: 1.** (697) organization, organisation -- (a group of people who work together) **ORGANIZATION: 3.** (39) administration, governance, governing body, establishment, brass, organization, organisation -- (the persons (or committees or departments etc.) who make up a body for the purpose of administering something; "he claims that the present administration is corrupt"; "the governance of an association is responsible to its members"; "he quickly became recognized as a member of the establishment") **BUSINESS: 1.** (1268) business, concern, business concern, business organization, business organisation -- (a commercial or industrial enterprise and the people who constitute it; "he bought his brother's business"; "a small mom-and-pop business"; "a racially integrated business concern") **GOVERNMENT: 1.** (504) government, authorities, regime -- (the organization that is the governing authority of a political unit; "the government reduced taxes"; "the matter was referred to higher authorities")

Table 5 continued.

	SECTOR: 2. (4) sector -- (a body of people who form part of society or economy; "the public sector") **POLICE: 1.** (34) police, police force, constabulary, law -- (the force of policemen and officers; "the law came looking for him") **INVESTOR: 1.** (3) investor -- (someone who commits capital in order to gain financial returns) **NEWSPAPER: 1.** (31) newspaper, paper -- (a daily or weekly publication on folded sheets; contains news and articles and advertisements; "he read his newspaper at breakfast") **SOCIAL GROUP: 1.** social group -- (people sharing some social relation)
LOCATION	**LOCATION: 1.** (992) location -- (a point or extent in space) **CELESTIAL BODY: 1.** (1) celestial body, heavenly body -- (natural objects visible in the sky) **WORLD: 2.** (232) universe, existence, creation, world, cosmos, macrocosm -- (everything that exists anywhere; "they study the evolution of the universe"; "the biggest tree in existence") **ACRE: 1.** (16) acre -- (a unit of area (4840 square yards) used in English-speaking countries) **GEOLOGICAL FORMATION: 1.** geological formation, formation -- ((geology) the geological features of the earth) **COMPASS POINT: 1.** compass point, point -- (any of 32 horizontal directions indicated on the card of a compass; "he checked the point on his compass")
FACILITY	**FACILITY: 1.** (35) facility, installation -- (a building or place that provides a particular service or is used for a particular industry; "the assembly plant is an enormous facility") **STRUCTURE: 1.** (24) structure, construction -- (a thing constructed; a complex construction or entity; "the structure consisted of a series of arches"; "she wore her hair in an amazing construction of whirls and ribbons") **SPACE STATION: 1.** space station, space platform, space laboratory -- (a manned artificial satellite in a fixed orbit designed for scientific research) **WAY: 6.** (88) way -- (any artifact consisting of a road or path affording passage from one place to another; "he said he was looking for the way out") **PORT: 1.** (5) port -- (a place (seaport or airport) where people and merchandise can enter or leave a country) **CAMPUS: 1.** (2) campus -- (a field on which the buildings of a university are situated) **TRACT: 1.** (3) tract, piece of land, piece of ground, parcel of land, parcel -- (an extended area of land)
GPE	**POLITICAL UNIT: 1.** (1) political unit -- (a unit with political responsibilities) **STATE: 1.** (354) state -- (the group of people comprising the government of a sovereign state; "the state has lowered its income tax") **STATE: 3.** (184) state, nation, country, land, commonwealth, res publica, body politic -- (a politically organized body of people under a single government; "the

Table 5 continued.

	state has elected a new president"; "African nations"; "students who had come to the nation's capitol"; "the country's largest manufacturer"; "an industrialized land") **NATION: 2.** (80) nation, land, country, a people -- (the people who live in a nation or country; "a statement that sums up the nation's mood"; "the news was announced to the nation"; "the whole country worshipped him") **POWER: 4.** (43) world power, major power, great power, power, superpower -- (a state powerful enough to influence events throughout the world) **REGIME: 1.** (2) government, authorities, regime -- (the organization that is the governing authority of a political unit; "the government reduced taxes"; "the matter was referred to higher authorities") **COUNTY: 2.** county -- (the largest administrative district within a state; "the county plans to build a new road") **POLITICAL SYSTEM: 1.** (3) political system, form of government -- (the members of a social organization who are in power) **SOCIETY: 1.** (88) society -- (an extended social group having a distinctive cultural and economic organization) **ADMINISTRATIVE DISTRICT: 1.** administrative district, administrative division, territorial division -- (a district defined for administrative purposes)

WordNet Features

The WordNet features employ the knowledge of the *WordNet-equivalent-concepts* and their lists of synonyms and hyponyms to decide whether a mention is of a certain type, i.e. in the synonym or hyponym set of certain *WordNet-equivalent-concepts*.

There are five features, each corresponding to an entity type (*is-a-TYPE*), each of which test whether the word is a part of the synonym or hyponym set of any of the *WordNet-equivalent-concepts* associated with that entity type. There are also features associated with each of the *WordNet-equivalent-concepts*, which test whether the word is in the synonym or hyponym set of that particular concept (*WN-eq-concept-hyp, WN-eq-concept-syn*). These latter features are necessary because it appeared that the manual association of the equivalent concepts with

entity types was not perfect, and experiments on the training set yielded better results if the more detailed features were present in addition to the five features for each entity type.

Since the initially-designed WordNet features would have taken a lot of time to run, an optimization was necessary. A preprocessing of WordNet extracted the lists with all holonyms, hypernyms, hyponyms, meronyms and synonyms of the *WordNet-equivalent-concepts* for each entity type, creating a file for each set, which was then loaded once. The features simply check whether the mention is in the respective list. With the optimization, the features work very fast. Although the plan is to employ all these sets as a part of features eventually, for this stage only the synonym and hyponym sets were used, as being an intuitive way of specifying the concepts.

Lexical Features

The lexical feature is the pair between the stem of the word and the WordNet sense of the word, as delivered by the word sense disambiguator. This feature is useful because some words are almost always of a certain type (e.g. *"company"*).

Syntactic Features

The part of speech of the word is the first syntactic feature used. The part of speech is taken from the output of the Brill part of speech tagger, which was applied to the text as part of preprocessing it. This feature is useful because it contains information about the number of the head of the noun phrase considered (e.g. NNS stands for plural noun).

The second group of syntactic features are the modifier ones. There is one feature that expresses whether the noun phrase considered is a modifier in another noun phrase (*is-modifier*), and there are five features (one corresponding to each entity type) that are true if the word appears as modifier in another noun phrase of that particular entity type (*modifier-to-TYPE*). These features are useful mainly to say which type the noun phrase does *not* belong to; if they are part of a mention with an entity type, it is most likely they will not share the type. They can also determine patterns of having modifiers of a certain type to mentions of a certain type (e.g. *"The New York Mets"*, where *"New York"* is a GPE and it modifies the whole NP, which is an ORGANIZATION).

The final syntactic feature is the apposition feature (*in-apposition-with*). It makes use of the fact that noun phrases in apposition are always coreferent, therefore of the same entity type. The feature value of the apposition for a mention is the entity type of the mention it is in apposition with.

Intelligent Context Features

The intelligent context set of features are an improvement on basic context features that use the stems of the words that are within a window of a certain size around the word. After studying the training data feature vectors, the conclusion was that most of the words in a normal context window were irrelevant, and a deeper analysis was needed to be able to make more intelligent context features.

The intelligent context features are divided into two categories. The first category contains features for all name, nominal, adjectival and pronominal modifiers to the head of the

mention that are within the parse tree of the mention (*all-modifiers*). For personal pronouns, the value of the feature is PERSON, while for the rest of the pronouns the value is the word itself. For named entities, the value of the feature is the entity type of the proper noun, as detected by the named entity recognizer applied to the text in the preprocessing phase. Finally, for common nouns and adjectives the value is the stem-sense pair. The second category of intelligent context features comprises one feature for each preposition that appears immediately before and immediately after the mention in its sentence (if they exists), with the lemma of the preposition as feature value.

In addition to this set of features, more features were created by combining them into pairs. Each pair contains two features from two different classes. For instance, there will be features like: *is-a-PERSON ~ in-apposition-with(*PERSON*)*.

All these features apply to the "true head" of a noun phrase, i.e. if the noun phrase is a partitive construction (*"five students"*, *"a lot of companies"*, *"a part of the country"*), the "true head" is the whole entity that the part was taken out of (*"students"*, *"companies"*, *"country"*), and the features are applied to that "true head" instead of the partitive head.

For combining the mention detection module with the BESTCUT coreference resolver, classifications for named entities and pronouns were also generated, by using the same set of features minus the WordNet ones (which only apply to nominal mentions). For the named entity classifier, the feature *named-entity-type* was added, as obtained by the named entity

recognizer. A list of all the markable mentions and their entity types was generated and presented as input to the BESTCUT resolver instead of the list of perfect mentions.

Note that this mention detection method does not obtain complete anaphoricity information, i.e. information about whether the mention refers to the same entity as a mention before it. Only the mentions that are a part of the five considered classes are treated as anaphoric and clustered, while the UNK mentions are ignored, even if an anaphoricity classifier might categorize some of them as anaphoric. This is in conformity with the annotation of the ACE corpus.

CHAPTER 4

BESTCUT— A GRAPH ALGORITHM FOR ENTITY TRACKING

4.1 The Problem of Entity Tracking

In texts, the same entities are mentioned multiple times. An **entity** is an object or a set of objects in the real world, while a **mention** is a textual reference to an entity[6]. To be able to automatically detect entities and their mentions in texts, a two-step procedure is followed:

> ***Step 1.*** Mentions of each entity are detected;
>
> ***Step 2.*** Mentions corresponding to the same entity are grouped together.

The first step is known as *the mention detection step*. This step is typically performed by first identifying where each mention starts and where it ends. This process is also known as mention boundary detection. After that, mentions are classified with respect to (a) their entity type or semantic class (e.g. PERSON, ORGANIZATION); (b) their mention type (e.g. name, nominal or pronominal) or (c) their genericity (e.g. general or specific). The second step is also performed in a sequence of two phases. Firstly, the likelihood that each possible pair of mentions refer to the same entity is evaluated. Secondly, based on these likelihood values, mentions are clustered together, each cluster corresponding to a different entity. This second step is also known as *entity tracking*.

The BESTCUT algorithm, described in this section, tackles only the second step of the procedure of automatically detecting entities and their mentions in texts. Chapter 3 of this thesis describes the first step of the procedure. The novelty introduced by BESTCUT consists of (1) the usage of a graph-based representation of mentions in the text and (2) a graph cutting method that identifies mentions that refer to the same entity. To illustrate the problem solved by BESTCUT and to exemplify its operation through a walk-through example, a text has been selected from the ACE corpus ([ACE Phase 2, 2003]). The text is illustrated in Figure 8(a), and the entities present in this excerpt and their mentions are shown in Figure 8(b).

Sen. **John McCain$_1$**, who sponsored the leading tobacco bill in **Congress$_2$**, said on **CBS$_3$**'s "Face the Nation," "**I$_1$**'m optimistic that **we$_4$** can get this done by this summer." Noting that the **White House$_5$** and public health advocates have complained that **his$_1$** bill isn't tough enough while the **industry$_6$** has said **it$_6$** cannot live with **his$_1$** bill, **McCain$_1$** said, "**I$_1$** think **we$_4$** may be well-positioned."	E_1 (PERSON): John McCain, I, his, his, McCain, I E_2 (ORGANIZATION): Congress E_3 (ORGANIZATION): CBS E_4 (PERSON): we, we E_5 (ORGANIZATION): White House E_6 (ORGANIZATION): industry, it
(a)	(b)

Figure 8. A text example from MUC-6 and the entities and mentions it contains.

When identifying mentions that refer to the same entity, we are inherently solving a coreference resolution problem. The problem of coreference resolution can also be described in terms of anaphors. In a text, anaphors are expressions that refer to some previously mentioned textual expressions. Therefore, a set of coreference chains can be formed, in which each anaphor is linked to its immediately preceding referent. Each coreference chain corresponds to an entity, and all expressions from the chain correspond to the entity mentions in the text. Because of this correspondence, entities that are mentioned only once can be

---

[6] This definition was introduced in [NIST, 2003].

thought of as coreference chains that have a single node. In this spirit, coreference chains with a single node contain no anaphors. For the example in Figure 8, the coreference chains are illustrated in Figure 9. Figure 9 shows that there are six different coreference chains, each corresponding to one of the entities listed in Figure 8(b). Coreference chains 2 and 3 have a single node. Additionally, expressions that belong to the same coreference chains (or mentions that corefer) are represented within the same type of graphical ellipse. For example, all expressions belonging to coreference chain 1 are represented within a double-lined ellipse.

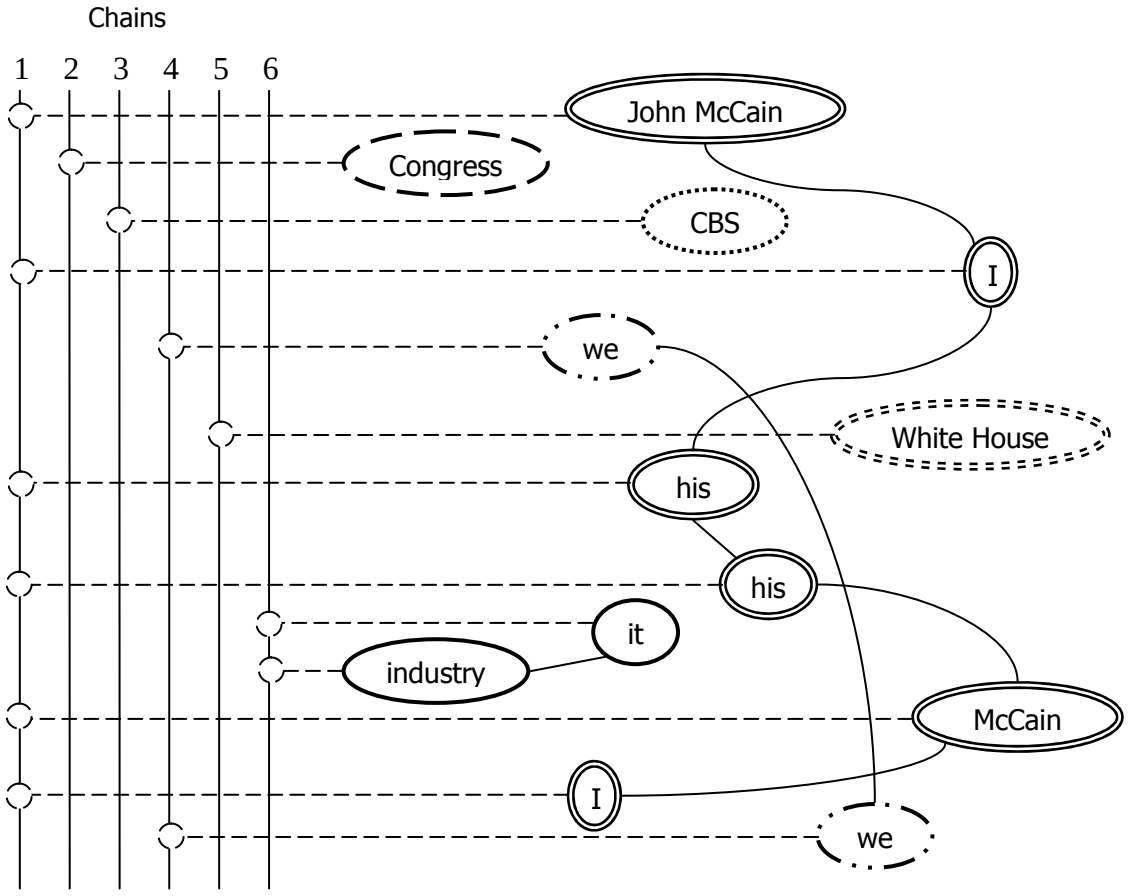


Figure 9. The coreference chains in the example.

Figures 8 and 9 illustrate on the same example the problem of entity tracking. In Figure 8, each entity is associated with the ordered list of its mentions in the text (and its semantic

category). In Figure 9, each entity is associated with a coreference chain, having ordered nodes that correspond to the coreferring expressions in texts. It can be noted that for each entity the mentions from the lists in Figure 8(b) and the nodes of the coreference chains from Figure 9 are identical. They correspond to two different representations of the same problem. Before the first ACE evaluations in 2000, the problem was known as the coreference resolution problem. Since 2002, the problem is known as the entity tracking problem.

4.2 The BESTCUT Algorithm for Coreference Resolution

In this thesis, the coreference space is represented in a novel way. The representation consists of a set of an undirected edge-weighted graphs in which the nodes are the mentions identified in the text, while the edges between nodes constitute the likelihood that the pair of nodes corefer. Additionally, in each graph, all mentions share the same semantic class. Figure 10 illustrates such a representation, which was generated for the text illustrated in Figure 8(a). The first graph represents possible coreference between mentions of persons, whereas the second graph represents possible coreference between mentions of organizations.

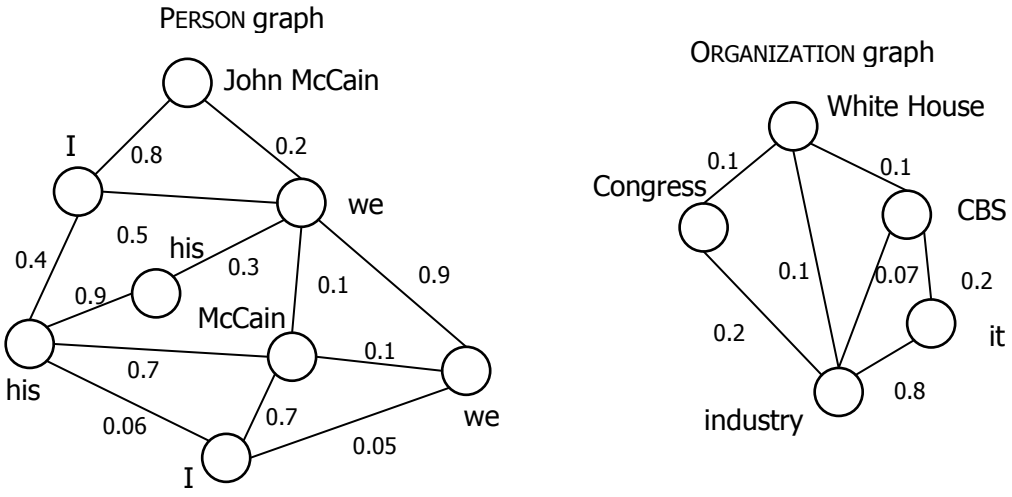


Figure 10. The graph representation of the coreference space
of the text illustrated in Figure 8.

To generate the graph corresponding to the coreference space of a text, we need to have access to two types of information: (1) we need to know all mentions identified in the text, such that each mention is assigned a node in the graph; and (2) we need to know the likelihood that each pair of mentions is coreferring. Moreover, whenever this likelihood exists, we need to have a measure of it, which is used for assigning the weight of the edge between the corresponding nodes in the graph. Access to the first form of information is granted by the input of the BESTCUT algorithm, which considers that all mentions in the text are identified prior to its operation. Access to the second form of information is provided by the first step of the BESTCUT algorithm.

BESTCUT

Input: Mentions of entities identified in the text.
Output: A graph partition, each subgraph corresponding to an entity. The subgraphs contain (1) nodes that represent coreferring mentions and (2) edges that correspond to the "strength" of the coreference between mentions.

Step 1. Classification. Decide which pairs of text mentions are coreferring, and with what confidence.
Step 2. Create a graph for each semantic category (PERSON, ORGANIZATION, LOCATION, FACILITY or GPE). The nodes of the graph are text mentions of the same semantic category. The graph edges and their weights are based on the generated results of Step 1.
Step 3. Graph-based clustering. For each graph generated at Step 2, partition it into subgraphs that best approximate the real entities.

Figure 11. The BESTCUT algorithm.

Step 1 of the BESTCUT algorithm decides which pairs of text mentions are coreferring. Step 2 generates graphs that may correspond to the graph representation of the coreference space in Figure 10. In the first graph from Figure 10 we have mentions of two entities: E_1 of type PERSON, mentioned by *"John McCain"*, *"I"*, *"his"*, *"his"*, *"McCain"* and *"I"*, and E_4 of type PERSON, mentioned by *"we"* and *"we"* later in the text. In the second graph we have

mentions corresponding to four different entities of type ORGANIZATION. These entities are $E_{2,}$ mentioned as *"Congress"*, $E_{3,}$ mentioned as *"CBS"*, $E_{5,}$ mentioned as *"White House"* and $E_{6,}$ mentioned as *"industry"* and *"it"*. All these entities and mentions are also illustrated in Figure 8. The same identifier of each entity from Figure 8 was used above to explain the entities referred in the graphs from Figure 10, which illustrates the result of Step 2 of BESTCUT. Finally, Step 3 of BESTCUT is detailed in Section 4.2.3.

The first step of the BESTCUT algorithm has been used by several other coreference resolution algorithms, among which the most notable is the algorithm reported in [Luo et al., 2004]. This algorithm uses a different representation of the coreference space, which is provided by the Bell tree. The first step of the BESTCUT algorithm simply adds a new set of features that enhance the classification accuracy.

The second step of BESTCUT generates the initial graphs, one graph per entity type. This representation of the coreference space is novel. Graph-based representations have been used recently in several NLP applications, but BESTCUT is the first algorithm to use a graph representation for coreference resolution, and this representation allowed to make coreference decisions in a global way rather than a local way. The graph-based representation of the coreference space used by BESTCUT, which was illustrated in Figure 10, is different from the representations illustrated in Figures 8 and 9, which are based on lists and chains or linked lists, respectively. This representation is superior because it encodes additional information. Not only can we gather the set of coreferring mentions as nodes in the graph, and their semantic type (since each graph corresponds to one semantic type), but we also

have an assessment of the coreference strength between each pair of mentions. The most important feature of graph-based representations of the coreference space stems from their ability to allow a global view of the coreference relations between mentions. This view leads to global optimization of the coreference decisions. Previous attempts to use a richer representation of the coreference space were based on Bell trees ([Luo et al., 2004]). However, the Bell tree representation allows only a partial search in the tree, guided by heuristics, because of the volume of the tree. This explains the superior accuracy of the BESTCUT algorithm when compared to state-of-the-art coreference algorithms.

The third step of BESTCUT is inspired by Min-Cut, the well-known graph-partitioning algorithm [Stoer and Wagner, 1994]. The aim of the Min-Cut algorithm is to partition a graph into two parts by cutting the weakest links between them. The main procedure of the algorithm chooses the cut with the minimum cut value (*cut-weight*) from a list of proposed cuts. The weight of the cut of a graph into two subgraphs is the sum of the weights of the edges crossing the cut. BESTCUT has a different way of calculating the *cut-weight*, which is detailed in Section 4.2.3.

The method was reported in the Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006) as [Nicolae and Nicolae, 2006]. One of the anonymous reviewers made the following comment: *"This paper provides a good solution to the problem of clustering entities after scoring the strength of individual potential coreference links. It assumes that we definitely want to do this process in two stages, which is in line with how most people do it, so it is quite useful in that respect. The approach seems to*

be a sound and useful way to partition the graph, given the weights from the first stage of classification -- definitely preferable to the greedy manner typically used for this. NLP involves graphs of all sorts, and I think there is a lot to gain from casting our problems in terms of graph problems that have already received a lot of attention, and this paper is an example of doing just this."

4.2.1 Coreference as Classification

The decision whether pairs of mentions from a text corefer or not is cast as a classification problem. This decision is used in Step 1 of the BESTCUT algorithm. The classification method that is used by BESTCUT is based on the maximum entropy model introduced by [Berger et al., 1996]. This model was discussed in detail in Section 3.2.1. [Luo et al., 2004] have shown how to use maximum entropy for coreference resolution. Based on the data seen, the maximum entropy model offers an expression for the probability that there exists coreference between a mention m_i and a mention m_j:

$$P(C \mid m_i, m_j) = \frac{e^{(\sum_k \lambda_k f_{kj}(m_i, m_j, C))}}{Z(m_i, m_j)}$$

Equation 2: Maximum entropy applied to coreference resolution by [Luo et al., 2004].

In Equation 2, $f_k(m_i, m_j, C)$ is a feature function in which C indicates whether mentions mi and mj are in coreference (C=1) or not (C=0). λ_k is its weight (a Lagrange multiplier), while $Z(m_i, m_j)$ is a normalizing factor that ensures that $P(C \mid m_i, m_j)$ is a probability (between 0 and 1).

There are several implementations of the maximum entropy method. In the experiments used for implementing BESTCUT, the *maxent*[7] toolkit was used. To extract features and to apply them, the training data provided by ACE[8] and MUC-6[9] was used. The features are detailed in Section 4.2.2. The training examples are created in the same way as [Luo et al., 2004] for all pairs of mentions of the same type: a training example for a pair of mentions consists of the pair's feature vector and the outcome (coreferent/noncoreferent) taken from the key files[10].

4.2.2 Feature Representation for Coreference Classification

The feature set used in BESTCUT consists of two types of features: (a) the feature set used by [Luo et al., 2004], reproduced in Table 6, and (b) a new set. However the feature representation has three main differences from [Luo et al., 2004]: 1) No combination between features was used, to prevent long running times on the large amount of ACE data, i.e. only single features were employed. 2) Through an analysis of the validation data, seven new features were implemented (Table 7). 3) As opposed to [Luo et al., 2004], who represented all numerical features quantized (all features in an interval having the same value), BESTCUT represents numerical feature values through a set of binary features corresponding to intervals. This transformation was necessary because the *maxent* tool requires binary features.

---

[7] This toolkit is available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

[8] The ACE-2 corpus is available at http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T11.

[9] The MUC-6 corpus is available at http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T13.

[10] The key files are the training files of the corpora annotated by humans with coreference information.

Table 6. Features used for coreference resolution by [Luo et al., 2004].

#	***Category***	***Feature name***	***Feature description***
1	Lexical	*exact_strm*	true if two mentions have the same spelling
2		*left_subsm*	true if one mention is a left substring of the other
3		*right_subsm*	true if one mention is a right substring of the other
4		*acronym*	true if one mention is an acronym of the other
5		*edit_dist*	editing distance between two mention strings
6		*spell*	pair of actual mention strings
7		*ncd*	number of different capitalized words in two mentions
8	Distance	*token_dist*	how many tokens two mentions are apart
9		*sent_dist*	how many sentences two mentions are apart
10		*gap_dist*	how many mentions in between the two mentions in question
11	Syntax	*POS_pair*	POS-pair of two mention heads
12		*apposition*	true if two mentions are appositive
13	Count	*count*	pair of numbers, each counting how many times a mention string is seen
14	Pronoun	*gender*	pair of attributes of {female, male, neutral, unknown}
15		*number*	pair of attributes of {singular, plural, unknown}
16		*possessive*	true if a pronoun is possessive
17		*reflexive*	true if a pronoun is reflexive

Table 7. New coreference features.

#	***Category***	***Feature name***	***Feature description***
1	Lexical	*head-match*	true if the two heads are identical
2		*type-pair*	for name—its type, noun—NOUN, pronoun—its spelling
3		*name-alias*	true if a mention is an alias of the other one
4	Syntax	*same-gov-category*	true if both mentions are covered by the same type of node (e.g. NP, VP, PP)
5		*path*	the parse tree path from m_2 to m_1
6		*coll-comm*	true if either mention collocates with a communication verb
7	Grammatical	*gn-agree*	true if the two mentions agree in gender and number

To elaborate on the second difference, each numerical feature from the [Luo et al., 2004] feature set translates into a set of binary features that model the mathematical relations *less than* and *greater than* and position the feature value in a linear space divided into a number of intervals. The intervals are defined by three numerical parameters: s*tart*, *step* and *count.*

The *start* parameter determines the initial point of the first interval, the *step* is the length of an interval and *count+1* is the number of intervals (binary features) considered. The initial points of the intervals, P_i, are computed in the following way:

$$P_i = start + step * i, \quad with\ 0 \le i \le count - 1$$

For exemplification, let us consider a numerical feature. Suppose the length of the interval in the [Luo et al., 2004] approach *N=10*, and *start=0*, *step=5*, *count=6* in the BESTCUT model. Table 8 details the set of binary features that replaced the numerical feature. Note: The number of intervals (binary features) depends on the value of the feature.

Table 8. Binary representation of a numerical feature.

Feature value	***[Luo et al., 2004] representation (one binary feature)***	***New representation (7 binary features)***
0	F_0	LTE_0, LTE_5, LTE_10, LTE_15, LTE_20, LTE_25
5	F_0	LTE_5, LTE_10, LTE_15, LTE_20, LTE_25
10	F_1	LTE_10, LTE_15, LTE_20, LTE_25
28	F_3	GT_25

In Table 8, **LTE_*X*** is true for a feature value *v* if $v \le X$, and **GT_*X*** is true for a feature value *v* if $v > X$. Only the features that had the value *true* were represented in the table.

4.2.3 Coreference as Graph-Based Clusterization

When a graph is cut into two different subgraphs, each subgraph also represents a cluster of nodes in the graph. This way of looking at clusterization is different than the classical methods of clusterization like k-means or hierarchical clustering— the BESTCUT graph representation does not need any knowledge about the number of clusters and their seeds, and it serves to represent a space that is not hierarchical.

The clusterization process described in this section takes place in Step 3 of the BESTCUT algorithm. Since Step 2 of BESTCUT generates graphs containing mentions, and some mentions in the graph may refer to one entity, whereas other mentions may refer to a different entity (or entities), the aim of this clusterization process is to identify all mentions that refer to the same entity. Such mentions are structured in a subgraph of the original graph. The clusterization method was inspired by the Min-Cut algorithm, which is detailed later in this section. Like the Min-Cut algorithm, the clusterization used in BESTCUT proposes several possible graph cuts, which are slightly different than those proposed by Min-Cut. The clusterization method used in BESTCUT also has a different way of computing the cut weight, and thus it selects the best graph cut differently. Furthermore, by considering a stopping model, the clusterization method used in BESTCUT is generalizing the Min-Cut algorithm.

In the example, reproduced in Figure 12(a) for convenience, the mentions grouped according to entity type are listed in Figure 12(b).

Sen. **John McCain$_1$**, who sponsored the leading tobacco bill in **Congress$_2$**, said on **CBS$_3$**'s "Face the Nation," "**I$_1$**'m optimistic that **we$_4$** can get this done by this summer." Noting that the **White House$_5$** and public health advocates have complained that **his$_1$** bill isn't tough enough while the **industry$_6$** has said **it$_6$** cannot live with **his$_1$** bill, **McCain$_1$** said, "**I$_1$** think **we$_4$** may be well-positioned."	PERSON: John McCain, I, his, his, McCain, I, we, we ORGANIZATION: Congress, CBS, White House, industry, it
(a)	(b)

Figure 12. Mentions grouped according to entity type for the example.

Each of these two groups then becomes a graph, as in Figure 13. The graph edges that had negligible weights have been eliminated from the representation for clarity.

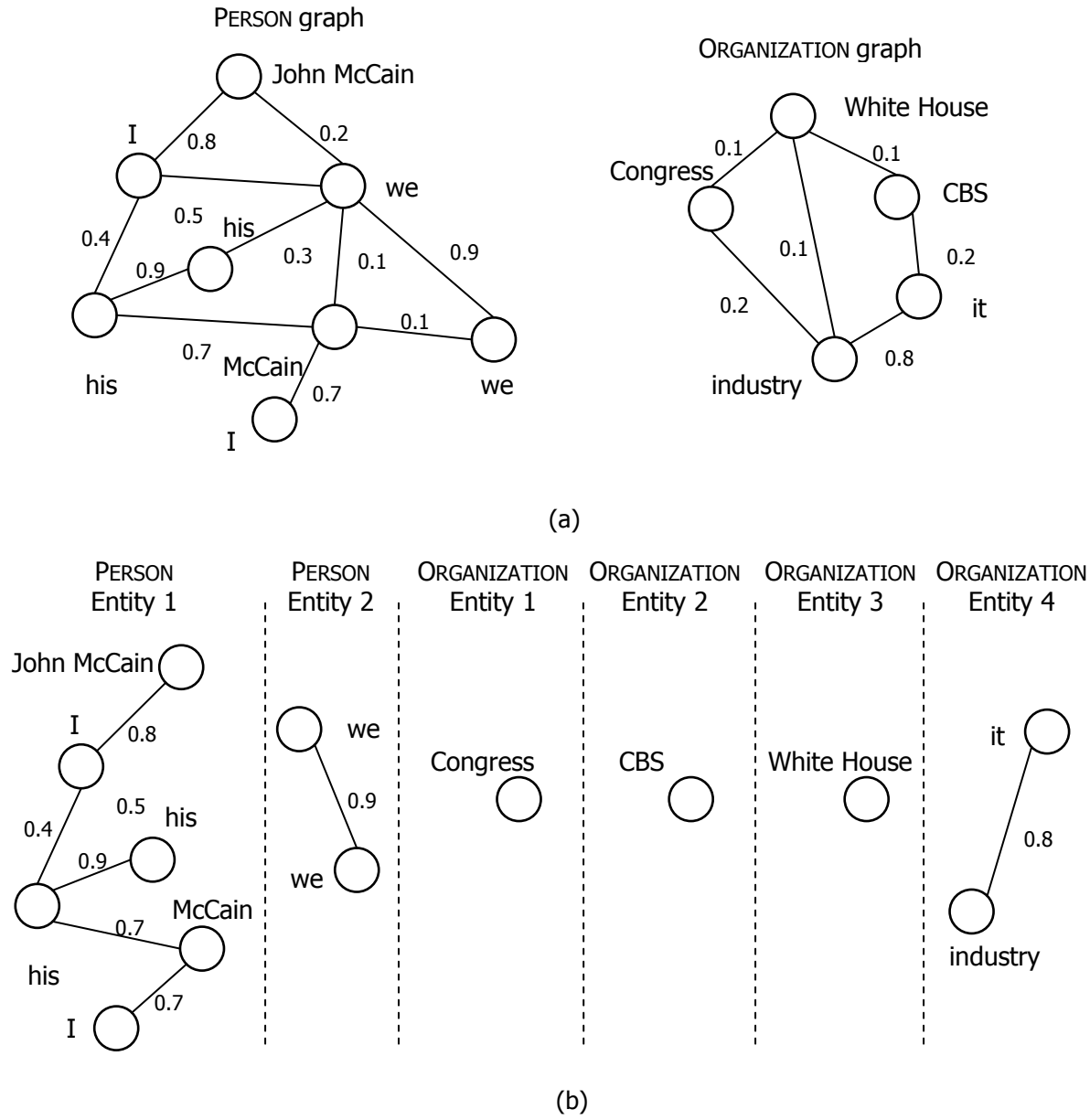


Figure 13. Mention graphs and their subgraphs corresponding to entities.

For each graph associated with an entity type, the goal is to obtain its partition into subgraphs that best approximate the real entities. BESTCUT cuts the graph recursively into smaller parts until it takes the decision to stop cutting.

The Min-Cut Algorithm

The Min-Cut algorithm, reported in [Stoer and Wagner, 1994], cuts a graph into its least connected two subgraphs. The algorithm proposes candidate cuts and selects from among them the one with the minimum cut value (*cut-weight*). The weight of the cut of a graph into two subgraphs is the sum of the weights of the edges crossing the cut. To compute a candidate cut, also known as a *cut-of-the-phase* of a graph, Min-Cut uses the algorithm called MINIMUMCUTPHASE, which is shown in Figure 14.

MINIMUMCUTPHASE

Input: Graph *G(V, E, w)*, a random start vertex *a*.
Output: A proposed cut *C* of the graph.

Step 1. Start with a set *A* containing the arbitrary vertex *a*.
Step 2. Add to the set of vertexes *A* the most tightly connected vertex *z*, that is:

$$z \notin A \quad such\ that \quad w(A,z) = \max\{w(A,y) \mid y \notin A\}$$

where $w(A,y) = \sum_{x \in A} w(x,y)$.

Step 3. Repeat Step 2 while $A \neq V$.
Step 4. Shrink *G* by merging the two vertexes added last to *A*.
Step 5. The cut *C* is the cut between the last vertex added and the rest of the vertexes.

Figure 14. The MINIMUMCUTPHASE procedure of the Min-Cut algorithm.

A subset *A* of the graph's vertexes grows starting with an arbitrary single vertex until *A* is equal to *V*. In each step, the vertex outside of *A* most tightly connected with *A* is added. At the end of each such phase, the two vertexes added last are merged, that is the two vertexes are replaced by a new vertex, and any edges from the two vertexes to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges. Edges joining the merged nodes are removed. The cut of *V* that separates the vertex added last from the rest of the graph is returned as *C*, the *cut-of-the-phase*.

The Min-Cut algorithm is presented in Figure 15. MINIMUMCUTPHASE is called repeatedly and the weights of the *cuts-of-the-phase* are compared to choose the minimum cut among them.

MINIMUMCUT

Input: Graph *G(V, E, w)*, an arbitrary vertex *a.*
Output: The minimum cut *MC* of the graph G.

Step 1. Call MINIMUMCUTPHASE on graph *G* and vertex *a* to propose a *cut-of-the-phase C*.
Step 2. If *C* **i**s lighter than the current minimum cut *MC*, store *C* as the current minimum cut *MC.*
Step 3. Repeat Steps 1-2 until graph *G* has only one vertex left.

Figure 15. The MINIMUMCUT procedure of Min-Cut.

The lightest of these *cuts-of-the-phase* is the result of the algorithm, the desired minimum cut.

Clusterization in BESTCUT

The Min-Cut algorithm was adapted for coreference and generalized in BESTCUT. The clustering algorithm receives as input a weighted graph associated to an entity type, as described before, and outputs the list of subgraphs (entities) created from its nodes (mentions). The BESTCUT algorithm is detailed in Figure 16. The method employs a queue to keep track of the subgraphs that need to be processed. Initially, the queue contains the input graph, and at each iteration the first graph is removed from the queue and processed— a best cut is proposed for it by ProposeCut. In case StopTheCut decides that the cut must be performed on the subgraph, the two sides of the cut are added to the back of the queue; if the graph is well connected and breaking the graph into two parts would lessen the performance, the current graph will be used to create a single entity. The algorithm ends when the queue becomes empty.

BESTCUT

Input: Graph G_i, a graph associated with a certain entity type.
Output: The set of entities in the graph.

Step 1: Start with a graph queue containing graph G_i.
Step 2: Extract the first graph in the queue, G.
Step 3: Propose a cut C for graph G by calling ProposeCut.
Step 4: Decide through the stop model if cutting G is preferable to keeping it whole. If G has to be cut, the cut is performed and the two resulting subgraphs are added to the back of the queue. If G must remain whole, it is added to the entity set as a new entity.
Step 5: Repeat Steps 2-4 until the queue becomes empty.

Figure 16. The BESTCUT clusterization method.

The results of applying the BESTCUT method on the ORGANIZATION graph are illustrated in Figure 17, in which the subgraphs have been annotated in the order of their appearance in the queue.

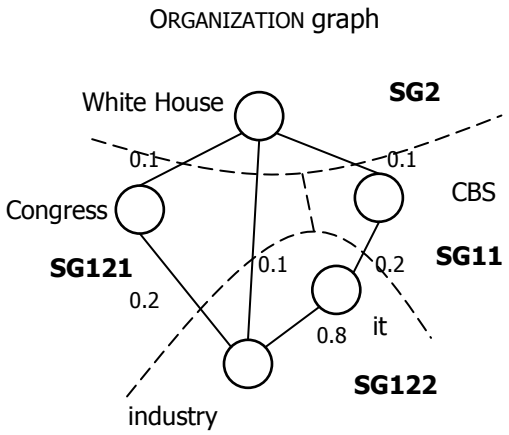


Figure 17. Final partitions obtained by BESTCUT on the ORGANIZATION graph, with the subgraphs annotated.

The evolution of the queue on the ORGANIZATION graph in the example is illustrated in Figure 18.

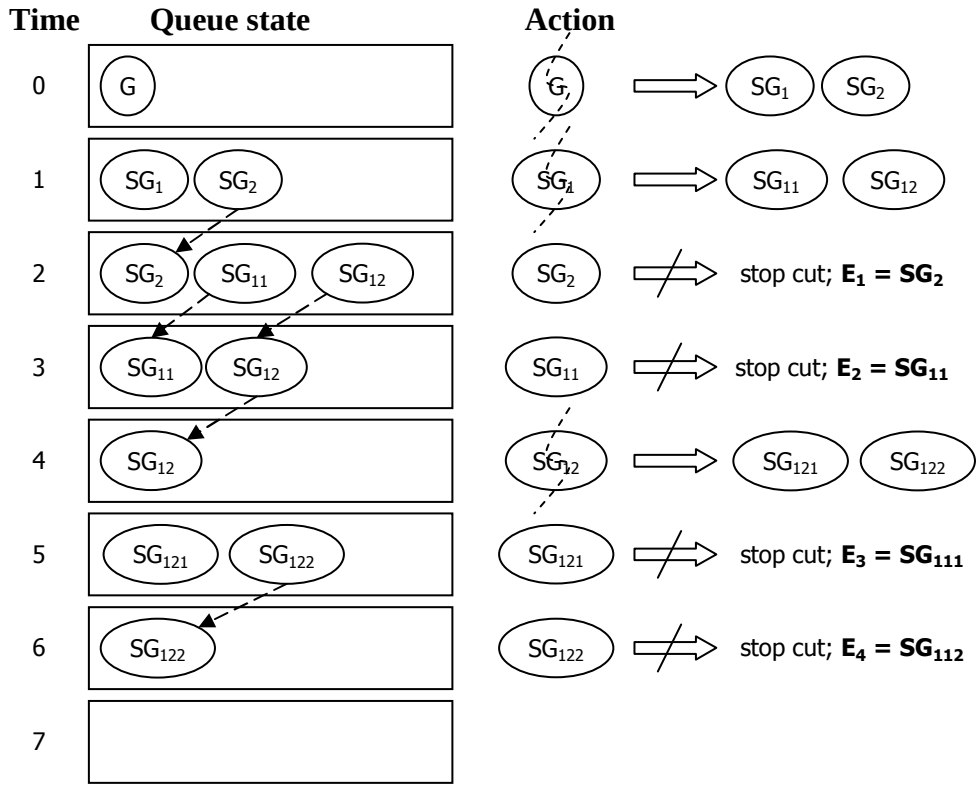


Figure 18. The evolution of the graph queue in applying the BESTCUT algorithm on the ORGANIZATION graph.

An important observation from Min-Cut that was also considered by BESTCUT is that in a graph ***G=(V, E)*** the number of all possible cuts is a very large number, given by the expression in Equation 3.

$$NC = \binom{|V|}{1} + \binom{|V|}{2} + \ldots \binom{|V|}{|V|/2} = 2^{|V|-1} + \frac{\frac{|V|!}{(|V|/2)!^2}}{2} - 1$$

Equation 3: The number of all possible cuts in a graph.

In Min-Cut and BESTCUT, the number of cuts considered for a graph ***G(V, E)*** is ***NC=|V|-1***. This is an obvious improvement on computation time.

Proposing Graph Cuts

At each iteration, for the first graph in the queue the algorithm proposes a best cut chosen from a set of *cuts-of-the-phase*. To obtain a *cut-of-the-phase,* as presented in Figure 19, the working set starts with a random first vertex. The procedure keeps adding to this set the most tightly connected vertex until there are no vertexes left in the graph. The two vertexes added last are merged and the cut that separates the last vertex from the rest of the graph is returned. If the *cut-of-the-phase* has a higher *cut-weight* than the current best cut, it becomes the new best cut. Note: Each *cut-of-the-phase* modifies the graph by merging two vertexes. The merging of the two vertexes is done, as in Min-Cut, by replacing the two vertexes with a new vertex, and replacing any edges from the two vertexes to a remaining vertex with an edge weighted by the sum of the weights of the previous two edges. Edges joining the merged nodes are removed.

ProposeCutPhase

Input: Graph *G* for which to propose a *cut-of-the-phase*.
Output: The *cut-of-the-phase C* proposed.

Step 1. Start with a set *A* containing the first vertex in the graph *G*.
Step 2. Add to the set of vertexes *A* the most tightly connected vertex z, that is:

$$z \notin A \quad such\ that \quad z = \arg\max_{y \notin A} w_a(A, y)$$

where $w_a(A, y) = \frac{1}{|A|} \sum_{x \in A} w(x, y)$.

Step 3. Repeat Step 2 while $A \neq V$.
Step 4. Shrink *G* by merging the two vertexes added last to *A*.
Step 5. The cut *C* is the cut between the last vertex and the rest of the vertexes.

Figure 19. The ProposeCutPhase procedure of BESTCUT clusterization.

ProposeCutPhase is analogous to the MINIMUMCUTPHASE Min-Cut procedure in Figure 14, but the difference is that the most tightly connected vertex, *z*, is found using a different expression:

$$z = \arg\max_{y \notin A} w_a(A, y)$$

where $w_a(A, y) = \frac{1}{|A|} \sum_{x \in A} w(x, y)$.

This new way to compute the weight $w_a(A, y)$ is normalized with the size of *A*, because $w(x, y)$ are accumulated weights which can be larger than 1.

Figure 20 presents the *cuts-of-the-phase* obtained in the first iteration on the initial ORGANIZATION graph.

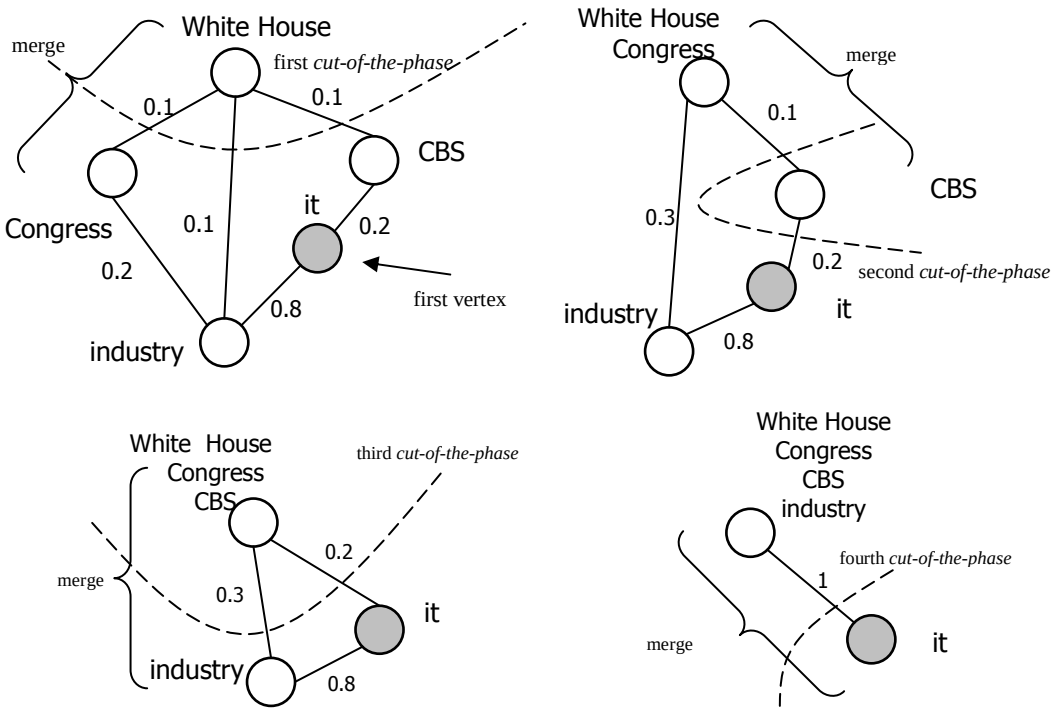


Figure 20. *Cuts-of-the-phase* in the first iteration on the ORGANIZATION graph.

Computing the *Cut-Weight*

At each iteration, the cut of the graph must be done in the way that ensures the best approximation of entities in the end. The final partition should preserve in the same cluster the mentions that are:

a) strongly connected on average;

b) maximally strongly connected.

To this end, the *cut-weight* used by the Min-Cut algorithm (the sum of the weights crossing the cut) was adapted to the coreference problem. To decide which cut to select as best cut among the *cuts-of-the-phase*, the *cut-weight* used is the average number of mentions that are correctly placed in their set. For each mention, two scores of correctness of its assignment to its subgraph of the cut are computed:

a) the average correctness;

b) the maximum correctness.

The *Average Correctness (AC)* score measures if the *average* of the weights of the edges connecting the mention to the rest of the mentions in its subgraph is higher than the *average* of the weights of the edges connecting the mention to the mentions of the other subgraph.

The *Maximum Correctness (MC)* score measures if the *maximum* of the weights of the edges connecting the mention to the rest of the mentions in its subgraph is higher than the *maximum* of the weights of the edges connecting the mention to the mentions of the other subgraph.

Note: For subgraphs containing only one mention, the average and maximal weight connecting the mention to its subgraph are both considered 0.5.

The *cut-weight* is computed by the algorithm in Figure 21, and its formal expression is shown in Equation 4. Starting from 0, the *AC* and *MC* scores are incremented with 1 for each mention in the graph when the mention is more tightly connected to its subgraph than it is with the other side of the cut, in an average or a maximal way respectively.

cut-weight

Input: The graph *G* that is being cut, cut *C=(S, T)* for which the weight needs to be computed.
Output: The *cut-weight* of cut *C*.

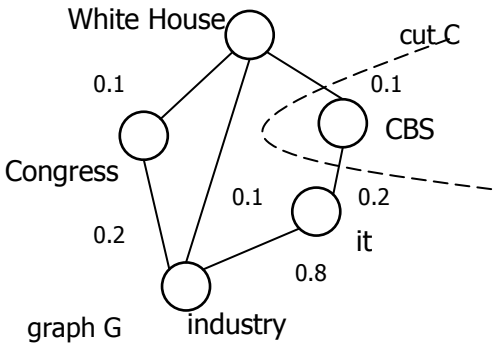
Step 1. Start with *corrects-avg* and *corrects-max* with value 0.
Step 2. For each mention *m* in graph G do:
 * increment *corrects-avg* if the *AC* score of mention *m* is 1;
 * increment *corrects-max* if the *MC* score of mention *m* is 1.
Step 3. The *cut-weight* is the average between *corrects-avg* and *corrects-max*).

Figure 21. Calculating the *cut-weight* for a cut ***C*** of a graph ***G***.

$$cut-weight(G,C)=\frac{\sum_{m\in G} AC(m,C)+\sum_{m\in G} MC(m,C)}{2}$$

Equation 4: The formal expression of the *cut-weight* for a cut ***C*** of a graph ***G***.

Here is an illustration of how the algorithm calculates a *cut-weight* for the second *cut-of-the-phase* in Figure 20.



$$\sum_{m \in G} AC(m,C) = AC("White\ House",C) + AC("Congress",C) + AC("CBS",C) + AC("it",C) + AC("industry",C) = 0+1+1+1+1 = 4$$

$$\sum_{m \in G} MC(m,C) = MC("White\ House",C) + MC("Congress",C) + MC("CBS",C) + MC("it",C) + MC("industry",C) = 0+1+1+1+1 = 4$$

$$cut-weight(G,C) = \frac{4+4}{2} = 4$$

Tables 9 and 10 detail the computing of the *Average Correctness*, *Maximum Correctness* and *cut-weights* for the four *cuts-of-the-phase* presented in Figure 20.

Table 9. Calculating the *Average Correctness* score for graph ***G***.

	Average Correctness					
cut	"White House"	"Congress"	"CBS"	"it"	"industry"	**SUM**
1	1	0	0	1	1	3
2	0	1	1	1	1	4
3	1	0	0	1	1	3
4	1	1	0	1	0	3

Table 10. Calculating the *Maximum Correctness* score for graph ***G*** and the *cut-weights* for the four *cuts-of-the-phase*, based on the *AC* and *MC* scores.

	Maximum Correctness						
cut	"White House"	"Congress"	"CBS"	"it"	"industry"	**SUM**	***cut-weight***
1	1	1	1	1	1	5	**4**
2	0	1	1	1	1	4	**4**
3	0	0	0	1	1	2	**2.5**
4	1	1	0	0	0	2	**2.5**

Selecting the Best Cut of the Graph

The ProposeCut algorithm (illustrated below, in Figure 22) returns as best cut the *cut-of-the-phase* with the highest *cut-weight*. At each iteration, a cut proposed by ProposeCutPhase is compared to the current best cut, and if it has a greater score than the best cut it becomes the new current best cut. ProposeCut returns a cut of the graph obtained with an algorithm similar to Min-Cut's procedure MINIMUMCUT.

```
ProposeCut

Input: Graph G for which to propose a best cut.
Output: A proposed best cut BC.

Step 1. Call ProposeCutPhase on graph G to propose a cut-of-the-phase C.
Step 2. If C has a greater cut-weight than the current best cut BC,
        store C as the new current best cut BC.
Step 3. Repeat Steps 1-2 until graph G has only one vertex left.
```

Figure 22. The ProposeCut algorithm.

In the example considered, the first two *cuts-of-the-phase* in Figure 20 are equal candidates, both having the *cut-weight* equal to 4 (see Table 10). The algorithm picks as best cut the first *cut-of-the-phase* encountered at iteration 1, which is illustrated below in Figure 23. In the next iteration, each side of this cut goes through the same processing as the initial graph.

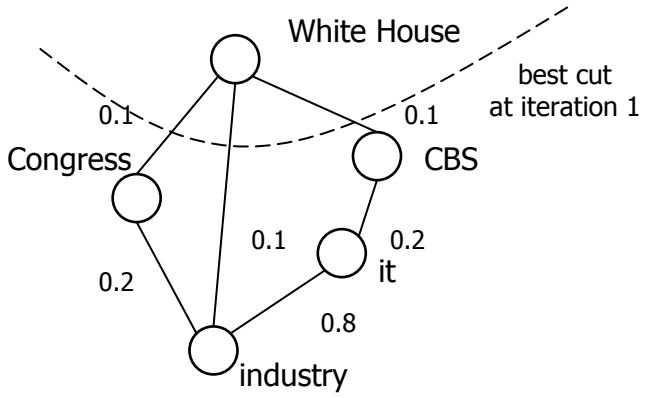


Figure 23. The best cut at iteration 1 for the example considered.

The Stopping Criterion for Graph Cutting

An additional learning model was trained to decide if cutting a set of mentions is desirable over keeping the mentions together. The cut stopping decision has been implemented as an SVM-based classification ([Cortes and Vapnik, 1995]). The model was optimized to maximize the ECM-F score[11]. The features for stopping the cut are presented in Table 11, where:

- *G* is the current graph before the cut;
- *S* is the larger part of the cut and *T* the smaller one;
- *S.V* and *T.V* are the sets of vertexes in *S* and *T* respectively;
- *S.E* and *T.E* are the sets of edges in *S* and *T* respectively;
- *C.E* is the set of edges crossing the cut.

The model was trained using 10-fold cross-validation on the ACE training set.

[11] As introduced by [Luo et al., 2004].

Table 11. The features for stopping the cut.

#	Feature	Description
1	*st_ratio*	|S.V| / |T.V| - the proportion of the cut
2	*ce_ratio*	|C.E| / |G.E| - the proportion of the cut from the entire graph
3	*c_min*	min(C.E) – the smallest edge crossing the cut
4	*c_max*	max(C.E) – the largest edge crossing the cut
5	*c_avg*	avg(C.E) – the average of the edges crossing the cut
6	*c_hmean*	hmean(C.E) – the harmonic mean of the edges crossing the cut
7	*c_hmeax*	hmeax(C.E) – a variant of the harmonic mean. hmeax(C.E) = 1 – hmean(C.E') where each edge from E' has the weight equal to 1 minus the corresponding edge from E
8	*lt_c_avg_ratio*	how many edges from the cut are less than the average of the cut (as a ratio)
9	*lt_c_hmean_ratio*	how many edges from the cut are less than the harmonic mean of the cut (as a ratio)
10	*st_avg*	avg(S.E + T.E) – the average of the edges from the graph when the edges from the cut are not considered
11	*g_avg*	avg(G.E) – the average of the edges from the graph
12	*st_wrong_avg_ratio*	how many vertexes are in the wrong part of the cut using the Average Correctness measure (as a ratio)
13	*st_wrong_max_ratio*	how many vertexes are in the wrong part of the cut using the Maximum Correctness measure (as a ratio)
14	*lt_c_avg_ratio< st_lt_c_avg_ratio*	1 if $r_1 < r_2$, 0 otherwise (r_1 is the ratio of the edges from *C.E* that are smaller than the average of the cut, r_2 is the ratio of the edges from *S.E + T.E* that are smaller than the average of the cut)
15	*g_avg > st_avg*	1 if the avg(G.E) > avg(S.E+T.E), and 0 otherwise

In order to learn when to stop the cut, positive and negative examples were generated from the training files. Each training example is associated with a certain cut *(S, T)*. Since the goal is to learn a stop function, the positive examples must be examples that describe when the cut must not be done, and the negative examples are situations when the cut must be performed. Formally, let $E = \{e_j: 1..m\}$ be the list of entities and $e_j = \{m_{j1}, m_{j2}, \ldots m_{jk}\}$ the list of mentions that refer to e_j. A negative example is generated for each pair $(S=\{e_i\}, T=\{e_j\}), i \neq j$ (each

entity must be separated from any other entity) and ($S=\{e_i\}$, $T=E\backslash S$) (each entity must be separated from all other entities together). To generate positive examples, the model simulates cutting a graph corresponding to a single entity e_j. Every partial cut of the mentions of e_j is considered as a positive example for the stop model.

(a)	(b)
Sen. **John McCain**$_1$, who sponsored the leading tobacco bill in **Congress**$_2$, said on **CBS**$_3$'s "Face the Nation," "**I**$_1$'m optimistic that **we**$_4$ can get this done by this summer." Noting that the **White House**$_5$ and public health advocates have complained that **his**$_1$ bill isn't tough enough while the **industry**$_6$ has said **it**$_6$ cannot live with **his**$_1$ bill, **McCain**$_1$ said, "**I**$_1$ think **we**$_4$ may be well-positioned."	E_1 (PERSON): John McCain, I, his, his, McCain, I E_2 (ORGANIZATION): Congress E_3 (ORGANIZATION): CBS E_4 (PERSON): we, we E_5 (ORGANIZATION): White House E_6 (ORGANIZATION): industry, it

Figure 24. Example revisited.

Figure 24 recalls the example we are studying. In this case, some negative examples for stopping the cut are: *($S=\{E_1\}$, $T=\{E_2\}$), ($S=\{E_1\}$, $T=\{E_3\}$), … ($S=\{E_5\}$, $T=\{E_6\}$), ($S=\{E_1\}$, $T=\{E_2, E_3, E_4, E_5, E_6\}$), … ($S=\{E_6\}$, $T=\{E_1, E_2, E_3, E_4, E_5\}$,* while positive examples for stopping the cut can be: *($S=\{$"John McCain"$\}$, $T=\{$"I"$\}$), ($S=\{$John McCain$\}$, $T=\{$his$\}$), … ($S=\{$John McCain, I, his$\}$, $T=\{$McCain, I$\}$), … ($S=\{$industry$\}$, $T=\{$it$\}$).*

Caveat

Pronouns are not included in the BESTCUT initial graphs, because, since most features are oriented towards named entities and common nouns, the maximum entropy learning algorithm links pronouns with very high probability to many possible antecedents, of which not all are in the same chain. Thus, in the clusterization phase the pronouns would act as a bridge between different entities that should not be linked. To prevent this, the pronouns are solved separately (at the end of the BESTCUT algorithm) by linking them to their antecedent with the best coreference confidence.

4.3 Advantages and Disadvantages of BESTCUT

This thesis proposes a novel coreference clusterization method that takes advantage of the simplicity of graph algorithms. The approach is top-down and globally optimized, and takes into account cataphora resolution in addition to anaphora resolution. The resulting system compares favorably to two other implemented clusterization systems and achieves state-of-the-art performance on the ACE corpus on key and detected mentions.

In summary, the advantages of BESTCUT are that:

1. it is simple and intuitive from being a graph algorithm.
2. it is based on Min-Cut, proven correct by [Stoer and Wagner, 1994].
3. it is a globally optimized approach to clusterization.
4. it uses a more appropriate representation of the coreference space than tree-based methods.
5. it performs cataphora resolution.
6. with mention information available it achieves state-of-the-art performance.

The disadvantages of BESTCUT are that:

1. with no prior mention information it has poor performance.
2. it has a longer running time than Greedy methods.
3. it is limited by the errors introduced by the maximum entropy classification.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Detecting Mentions

The metrics used in this evaluation were the following ([MUC-6, 1995]):

$$\text{Precision } P = \frac{\textit{number of correctly solved instances}}{\textit{total instances discovered}}$$

$$\text{Recall } R = \frac{\textit{number of correctly solved instances}}{\textit{total instances in annotated file}}$$

$$\text{F-factor } F = \frac{2 * P * R}{P + R}$$

$$\text{Accuracy } A = \frac{\textit{number of correctly solved instances}}{\textit{total instances}}$$

The difference between precision and accuracy, at least in the case of entity type (multiple) outcomes, is that precision is calculated only for entities of one of the five known types, while accuracy also involves the UNK instances, which makes it less relevant, because the UNK types constitute the vast majority of the entity types in the documents.

Three types of experiments were performed. In the first type of experiment, the outcome for each training and testing example was the entity type of the mention; in the second, the outcome for each training and testing example was the genericity of the mention, whereas in

the third, the outcome in both training and testing examples was the pair entity type–genericity of the mention. In the latter case, the results were reported in three ways: (1) by considering only the correctness of the entity types obtained by the classifier and ignoring the genericity values of the pairs; (2) by considering only the correctness of the genericity values and ignoring the entity types of the pairs; and (3) by taking into account the correctness of the entity type–genericity pair outcomes. The experiments were performed on the ACE Phase 2 [ACE Phase 2, 2003] corpus. The results for each type of outcome: entity type, genericity, and entity type–genericity pair, are shown in the following tables.

Table 12. Results for mention detection with entity type as outcome.

	P%	**R%**	**F%**
Overall	80.86	73.93	**77.24**
PERSON	86.69	85.99	86.34
ORGANIZATION	71.5	60.16	65.34
LOCATION	45.35	58.21	50.98
FACILITY	60	43.57	50.48
GPE	82.22	68.01	74.45
Accuracy = 91%			

The experimental results presented in Table 12 constitute the best performance obtained in classifying mentions into entity classes. This configuration, which takes into account only entity types as outcomes, outperforms the configuration that also employs genericity information as outcomes, whose results will be shown in Table 14.

Table 13. Results for mention detection with genericity as outcome.

	P%	**R%**	**F%**
Overall	93.04	93.04	**93.04**
GENERIC	38.79	38.19	38.49
SPECIFIC	96.26	96.35	96.31
Accuracy = 93.04%			

The high results reported in Table 13 are misleading because of the fact that specific mentions constitute the majority of all mentions in a document, therefore a baseline system which would classify all mentions as specific would have very high performance.

Table 14. Results for mention detection for entity type–genericity pairs as outcome, only considering entity type.

	P%	**R%**	**F%**
Overall	79.33	70.86	**74.85**
PERSON	86.04	84.51	85.26
ORGANIZATION	68.75	55.89	61.66
LOCATION	60.38	37.21	46.04
FACILITY	61.22	33.52	43.32
GPE	71.77	65.44	68.46
Accuracy = 90%			

The configuration that uses as outcome entity type–genericity pairs (Table 14) has a worse performance than the one that does not consider genericity information. The observation that can be drawn from these two experiments is that genericity information, instead of helping, introduces noise into the classifier, and is an area that needs future work.

Table 15. Results for mention detection for entity type–genericity pairs as outcome, only considering genericity.

	P%	**R%**	**F%**
Overall	92.58	92.58	**92.58**
GENERIC	34.93	34.93	34.93
SPECIFIC	96.06	96.06	96.06
Accuracy = 92.58%			

The results for the experiment in which pairs are considered as outcome (Table 15), even if only genericity is taken into account for the scoring, are slightly worse than in the case of

genericity as outcome (Table 13). This shows that entity type also introduces noise in the genericity classifier, not only vice versa.

Table 16. Results for mention detection for entity type–genericity pairs as outcome, considering both the entity type and the genericity.

		P%	R%	F%
Overall		56.03	50.04	**52.86**
PERSON	GENERIC	39.15	36.82	37.95
	SPECIFIC	65.26	65.26	65.26
ORGANIZATION	GENERIC	16.66	25	20
	SPECIFIC	59.06	41.51	48.75
LOCATION	GENERIC	0	0	0
	SPECIFIC	58	36.71	44.96
FACILITY	GENERIC	21.05	21.05	21.05
	SPECIFIC	54.43	26.87	35.98
GPE	GENERIC	15.79	12	13.64
	SPECIFIC	66.37	61.54	63.86
Accuracy = 84.64%				

The results for taking the entity type–genericity pairs as outcome, and classifying according to both the entity type and the genericity, are the worst in performance. This happens because partially correct results are penalized. For instance, if a mention is a GENERIC PERSON, and the classifier detects it as a SPECIFIC PERSON, this is considered an error, whereas if the result considered only the entity type detected for the mention, this would be a correct instance.

5.2 Resolving Coreference

The clusterization algorithms implemented to evaluate in comparison with the BESTCUT method are [Luo et al., 2004]'s Belltree and Link-Best (best-first clusterization) from [Ng and Cardie, 2002]. The features used were described in Section 4.2.2. The experiments were performed on the ACE Phase 2 [ACE Phase 2, 2003] and MUC-6 [MUC-6, 1995] corpora.

Since the aim was to measure the performance of coreference, the metrics used for evaluation are the ECM-F [Luo et al., 2004] and the MUC P, R and F scores [Vilain et al., 1995].

In the first experiment, all three coreference clusterization algorithms were tested on the development-test set of the ACE Phase 2 corpus, first on true mentions (i.e. the mentions annotated in the key files), then on detected mentions (i.e. the mentions output by the mention detection system presented in Chapter 3), and finally without any prior knowledge of the mention types. The results obtained are tabulated in Table 17.

Table 17. Comparison of results between three clusterization algorithms on ACE Phase 2.

Clusterization Algorithm	***Mentions***	**ECM-F%**	**MUC *score***		
			MUC P%	**MUC R%**	**MUC F%**
BESTCUT	key	**82.7**	**91.1**	88.2	**89.63**
	detected	**73.0**	**88.3**	75.1	**81.17**
	undetected	41.2	**52.0**	82.4	**63.76**
Belltree [Luo et al., 2004]	key	77.9	88.5	89.3	88.90
	detected	70.8	86.0	76.6	81.03
	undetected	**52.6**	40.3	87.1	55.10
Link-Best [Ng and Cardie, 2002]	key	77.9	88.0	**90.0**	88.99
	detected	70.7	85.1	**77.3**	81.01
	undetected	51.6	39.6	**88.5**	54.72

As can be observed, when it has prior knowledge of the mention types BESTCUT performs significantly better than the other two systems in the ECM-F score and slightly better in the MUC metrics. The more knowledge it has about the mentions, the better it performs. This is consistent with the fact that the first stage of the algorithm divides the graph into subgraphs corresponding to the five entity types. If BESTCUT has no information about the mentions, its performance ranks significantly under the Link-Best and Belltree algorithms in ECM-F and MUC R. Surprisingly enough, the Belltree algorithm, a globally optimized algorithm, performs similarly to Link-Best in most of the scores.

Despite not being as dramatically affected as BESTCUT, the other two algorithms also decrease in performance with the decrease of the mention information available, which empirically proves that mention detection is a very important module for coreference resolution. Even with an F-score of 77.2% for detecting entity types, the mention detection system boosts the scores of all three algorithms when compared to the case where no information is available.

It is apparent that the MUC score does not vary significantly between systems. This only shows that none of them is particularly poor, but it is not a relevant way of comparing methods– the MUC metric has been found too indulgent by researchers ([Luo et al., 2004], [Baldwin et al., 1998]). The MUC scorer counts the common links between the annotation keys and the system output, while the ECM-F metric aligns the detected entities with the key entities so that the number of common mentions is maximized. The ECM-F scorer overcomes two shortcomings of the MUC scorer: (1) not considering single mentions and (2) treating every error as equally important ([Baldwin et al., 1998]), which makes the ECM-F a more adequate measure of coreference.

The second experiment evaluates the impact that the different categories of the added features have on the performance of the BESTCUT coreference system. The experiment was performed with a maxent classifier on the MUC-6 corpus converted into ACE format, and employed mention information from the key annotations. The baseline system for performing the comparison has only the [Luo et al., 2004] features. The results are tabulated in Table 18.

Table 18. Impact of feature categories on BESTCUT on MUC-6.

Model	**ECM-F%**	**MUC *score***		
		MUC P%	**MUC R%**	**MUC F%**
baseline	78.3	89.5	91.5	90.49
+grammatical	78.4	89.2	92.5	90.82
+lexical	83.1	92.4	91.6	92.00
+syntactic	**85.1**	**92.7**	**92.4**	**92.55**

From Table 18 it can be observed that the lexical features (*head-match*, *type-pair*, *name-alias*) have the most influence on the ECM-F and MUC scores, succeeded by the syntactic features (*same-governing-category*, *path*, *coll-comm*). Despite what intuition suggests, the improvement the grammatical feature *gn-agree* brings to the system is very small.

CONCLUSIONS

This thesis has presented two novel methods for entity detection and tracking. The problem of entity detection and tracking was split into two consecutive stages: mention detection and entity tracking (or coreference resolution). The method developed to solve the first stage focuses on obtaining the semantic class of nominal mentions, information from which the second stage benefits. The novelty of this method consists in employing semantic information from the WordNet lexical database ([Fellbaum, 1998]) by mapping the semantic classes of the mentions into WordNet concept hierarchies, and in the use of word sense disambiguation. The choice of the semantic class for each mention in a text is cast as a classification problem. The classification algorithm employed is a maximum entropy statistical model introduced by [Berger et al., 1996], and serves to classify all the mentions from a text into one of the five semantic classes (or entity types) introduced by [ACE Phase 2, 2003]: PERSON, ORGANIZATION, LOCATION, FACILITY and GPE. The second stage of the problem, coreference resolution, is solved through a graph-based algorithm named BESTCUT. The novelty introduced by BESTCUT consists of (1) the usage of a graph-based representation of mentions in the text and (2) a graph cutting method that identifies mentions that refer to the same entity. The representation consists of a set of an undirected edge-weighted graphs in which the nodes are the mentions identified in the text, while the edges between nodes constitute the likelihood that the pair of nodes corefer. The likelihood values are learned by applying a maximum entropy classifier on the pairs of mentions in the text. Additionally, in

each graph, all mentions share the same semantic class. Clustering these mentions into entities is done by repeatedly cutting these graphs until the subgraphs obtained best approximate the real entities. The experimental results are competitive with the best results obtained in previous research work, which is encouraging for future exploring of graph-based and semantic representation of text data.

BIBLIOGRAPHY

[ACE Phase 2, 2003] ACE Phase 2 Program Committee. 2003. *Entity Detection and Tracking – Phase 1; EDT and Metonymy Annotation Guidelines.* Version 2.5.1 20030502.

[Baker et al., 1998] C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. *The Berkeley FrameNet project.* In Proceedings of the COLING-ACL, Montreal, Canada.

[Baldwin et al., 1998] B. Baldwin, T. Morton, A. Bagga, J. Baldridge, R. Chandraseker, A. Dimitriadis, K. Snyder, and M. Wolska. 1998. *Description of the university of pennsylvania camp system as used for coreference.* In Proceedings of the 7th Message Understanding Conference (MUC-7).

[Bean and Riloff, 2004] D. Bean and E. Riloff. 2004. *Unsupervised Learning of Contextual Role Knowledge for Coreference Resolution.* In Proceedings of HLT-NAACL 2004.

[Berger et al., 1996] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. *A maximum entropy approach to natural language processing.* Computational Linguistics, 22 (1): 39--68, 1996.

[Bikel et al., 1997] D. M. Bikel, S. Miller, R. Schwartz and R. Weischedel. 1997. *Nymble: a High-Performance Learning Name-finder.* In Proceedings of the Conference on Applied Natural Language Processing, 1997.

[Bikel et al., 1999] D. M. Bikel, R. Schwartz and R. M. Weischedel. 1999. *An Algorithm that Learns What's in a Name.* In the Machine Learning Journal Special Issue on Natural Language Learning.

[Carreras et al., 2002] X. Carreras, L. Márques and L. Padró. 2002. *Named Entity Extraction using AdaBoost.* In Proceedings of CoNLL-2002, Taipei, Taiwan, 2002, pp. 167-170.

[Cortes and Vapnik, 1995] Cortes C., Vapnik, V. *Support vector machines*. Machine Learning, 20, pp. 273-297, 1995.

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

[Cunningham, ?] H. Cunningham. *Information Extraction, Automatic.* Available at http://gate.ac.uk/sale/ell2/ie/main.pdf

[Daelemans et al., 2001] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2001. *Timbl: Tilburg memory based learner, version 4.0, reference guide.* Technical report, University of Antwerp.

[Fellbaum, 1998] C. Fellbaum, editor. 1998. *WordNet - An Electronic Lexical Database*. The MIT Press.

[Florian et al., 2003] R. Florian, A. Ittycheriah, H.Jing and T. Zhang. 2003. *Named Entity Recognition through Classifier Combination.* In Proceedings of CoNLL-2003, Edmonton, Canada, 2003, pp. 168-171.

[Florian et al, 2004] R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X Luo, N. Nicolov, S. Roukos. 2004. *A Statistical Model for Multilingual Entity Detection and Tracking.* HLT-NAACL 2004.

[Ge et al., 1998] N. Ge, J. Hale, E. Charniak. 1998. *A Statistical Approach to Anaphora Resolution.* Workshop ACL-COLING 1998.

[Halliday and Hasan, 1976] M. Halliday and R. Hasan. 1976. *Cohesion in English.* London: Longman, 1976.

[Hobbs, 1979] J. Hobbs. 1979. *Coherence and Coreference.* Cognitive Science, Vol. 3, No. 1, pp. 67-90.

[Hobbs, 1986] J. Hobbs. 1986. *Resolving Pronoun References.* Readings in natural language processing Pages: 339 - 352 Year of Publication: 1986

[Hobbs, 2002] J. Hobbs. 2002. *"the"*. Proceedings, 2002 International Symposium on reference Resolution for Natural Language Processing, pp. 45-49, Alicante, Spain, June 2002.

[Ittycheriah et al., 2003] A. Ittycheriah, L. Lita, N. Kambhatla, N. Nicolov, S. Roukos, M. Stys. 2003. *Identifying and Tracking Entity Mentions in a Maximum Entropy Framework.* HLT-NAACL 2003.

[Luo et al., 2004] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla and S. Roukos. 2004. *A Mention-Synchronous Coreference Resolution Algorithm Based On the Bell Tree.* In Proceedings of ACL 2004.

[Marcus et al., 1993] M. P. Marcus, B. Santorini and M. A. Marcinkiewicz. 1993. *Building a Large Annotated Corpus of English: the Penn Treebank.* Computational Linguistics, 19:313-330.

[Mihalcea and Csomai, 2005] R. Mihalcea and A. Csomai. *SenseLearner: Word Sense Disambiguation for All Words in Unrestricted Text.* In Proceedings of the 43nd

Annual Meeting of the Association for Computational Linguistics, companion volume (ACL 2005), Ann Arbor, MI, June 2005.

[Miller, 1995] G. A. Miller. 1995. *WordNet: a lexical database for English.* Communications of the ACM. Volume 38, Issue 11 (November 1995), Pages: 39 – 41, Year of Publication: 1995

[Mitkov, 1998] R. Mitkov. 1998. *Robust pronoun resolution with limited knowledge.* ACL-COLING 1998.

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

[Muller et al., 2002] C. Muller, S. Rapp, M. Strube. 2002. *Applying Co-Training to Reference Resolution.* In Proceedings of ACL 2002.

[Ng, 2004] V. Ng. 2004. *Learning Noun Phrase Anaphoricity to Improve Conference Resolution: Issues in Representation and Optimization.* In Proceedings of ACL 2004.

[Ng, 2005] V. Ng. 2005. *Machine Learning for Coreference Resolution: From Local Classification to Global Ranking.* In Proceedings of ACL 2005.

[Ng and Cardie, 2002] V. Ng, C. Cardie. 2002. *Improving Machine Learning Approaches to Coreference Resolution.* In Proceedings of ACL 2002.

[Ng and Cardie, 2003a] V. Ng, C. Cardie. 2003. *Bootstrapping Coreference Classifiers with Multiple Machine Learning Algorithms.* In Proceedings of EMNLP 2003.

[Ng and Cardie, 2003b] V. Ng, C. Cardie. 2003. *Weakly Supervised Natural Language Learning Without Redundant Views.* In Proceedings of HLT-NAACL 2003.

[Nicolae and Nicolae, 2006] C. Nicolae and G. Nicolae. 2006. *BESTCUT: A Graph Algorithm for Coreference Resolution.* In Proceedings of EMNLP-06 -- 2006 Conference on Empirical Methods in Natural Language Processing, Sydney, Australia, 2006.

[Palmer et al., 2005] M. Palmer, D. Gildea and P. Kingsbury. 2005. *The Proposition Bank: An Annotated Corpus of Semantic Roles.* Computational Linguistics, 31(1):71-106, 2005.

[Soon et al., 2001] W. M. Soon, H. T. Ng, D. C.Y. Lim. 2001. *A Machine Learning Approach to Coreference Resolution of Noun Phrases.* In Computational Linguistics archive Volume 27, Issue 4 (December 2001).

[Stoer and Wagner, 1994] M. Stoer and F. Wagner. 1994. *A simple min cut algorithm.* In Proceedings of the 1994 European Symposium on Algorithms, Jan van Leeuwen, ed. Springer-Verlag, New York, pp. 141–147.

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

[Vilain et al., 1995] M. Vilain, J.Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. *A model-theoretic coreference scoring scheme.* In Proceedings of the Sixth Message Understanding Conference (MUC-6), pages 45–52.

[Yang et al., 2003] X. Yang, G. Zhou, J. Su, C. L. Tan. 2003. *Coreference Resolution Using Competition Learning Approach.* In Proceedings of ACL 2003.

VITA

Cristina Nicolae was born on December the 16th, 1978 in Bucharest, the capital city of Romania. Her loving parents instilled in her the interest for computers from an early age. She followed their steps in the computer science field by going to the Politehnica University of Bucharest and studying to become an engineer in the Computer Science department. She specialized in Artificial Intelligence and Computer Graphics, and earned her Engineer degree in 2002 for her work in the automatic summarization of texts.

In her freshman year, Cristina met Gabriel, another student attending the same university, whom she married in 2000. Together, they relocated to Dallas, Texas in the fall of 2003, and they have been working and studying together at the Human Language Technology Research Institute of the University of Texas at Dallas ever since.