Entities and Identities: Named Entity Processing with cultural Knowledge

Anna Lisa Gentile
Dipartimento di Informatica
Dottorato di Ricerca in Informatica XXII ciclo
Università degli Studi di Bari "Aldo Moro"
Via E. Orabona, 4 - 70125 Bari, Italy
al.gentile@di.uniba.it

S.S.D.: INF/01

Supervisor: Prof. Giovanni Semeraro

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

February 2010

Credits

This dissertation was typeset using these open-source programs:

- TeXShop LaTeX Editor
  available at: `http://www.uoregon.edu/~koch/texshop/`
- MacTex Distribution
  available at: `http://www.tug.org/mactex/`

Thesis Supervisor

Prof. Giovanni Semeraro

Chairperson of the Supervisory Committee

Member of the Supervisory Committee

Member of the Supervisory Committee

Submitted *February 2010*

It is said that though we have all found out that there are no unicorns, of course there might have been unicorns. Under certain circumstances there would have been unicorns.
[...]
Perhaps according to me the truth should not be put in terms of saying that it is necessary that there should be no unicorns, but just that we can't say under what circumstances there would have been unicorns.

Kripke (1972)

Contents

List of Figures

Abstract

Natural Language is a mean to express and discuss about concepts, objects, events, i.e. it carries semantic contents. Reading a written text implies the comprehension of the information that words are carrying. Comprehension is an intrinsic capacity for a human, but not for a machine. One of the ultimate roles of Natural Language Processing techniques is identifying the meaning of the text, providing effective ways to make a proper linkage between textual references and real world objects, thus enabling machines to have a bit of the understanding which is proper of a human.

A proper name is a word or a list of words that refers to a real world object. Linguistic Expressions with the same reference may have different senses, so it is necessary to disambiguate between them.

Natural Language Processing (NLP) operations include text normalization, tokenization, stop words elimination, stemming, Part Of Speech tagging, lemmatization. Further steps, such as Word Sense Disambiguation (WSD) or Named Entity Recognition (NER), are aimed at enriching texts with semantic information. Named Entity Disambiguation (NED) is the procedure that solves the correspondence between real-world entities and mentions within text. One of the ultimate goals of NLP techniques is to identify the meaning of the text, providing effective ways to make a proper linkage between textual references and real world objects. The thesis addresses the problem of giving a sense to *proper names* in a text, that is the problem of automatically associating words representing *Named Entities* with their *identities*, that is unique real world objects. Also, the thesis copes with the problem of lack of training and testing data for such a task.

Proposed approaches automatically associate each entity in a text with

a unique identifier, a URI from Wikipedia[1], which is used as an "entity-provider".

The main contribution consists of proposing knowledge based approaches for NED, which do not requires training data. Specifically the thesis proposes two solutions:

- a completely knowledge-based algorithm for NED, exploiting Wikipedia data
- a Semantic Relatedness (SR) approach for the NED task: SR scores are obtained by a graph-based model over Wikipedia

The first solution has been tested for italian language: due to lack of italian testing data for such task, the thesis shows a method to automatically build a testbed dataset from Wikipedia. The second solution has been tested over an goldstandard dataset for NED: the proposed algorithm achieves results competitive with the state of the art.

Both suggested solutions are completely knowledge-based, with the advantage that no training data is needed: indeed, manually annotated data for this task is not easily available and acquiring such data can be expensive.

[1] http://www.wikipedia.org

Chapter 1

Introduction

In ordinary language it is possible to express the same concept with several different words, *synonyms*, but also many words, called *polysemic words*, have more than one possible meaning (as can be seen looking for a word in a dictionary). The properties that indicates such phenomenon are called respectively *Synonymy* and *Polysemy* and are properties for words. On the other hand, *ambiguity* is a property of text and it is due to how the speaker or writer intends to use a polysemous word. Polysemy indicates a potential ambiguity, which can be removed exploiting the context of usage. Word Sense Disambiguation (WSD) refers to the resolution of lexical semantic ambiguity and its goal is to attribute the correct sense to a word used in a given context.

Proper names are special words within the language, which own a special status. They have been considered by some philosophers, e.g. Ziff (1960), as not being part of language in some sense. Nevertheless, they have the same two properties described for common words: *Synonymy*, that for proper names is addressed as *Name Variation* and *Polysemy*, which is referred as *Name Ambiguity*. Exploiting this parallelism, the terms *meaning* or *sense*, when referred to a proper name, can be used to indicate the real world object designated by a proper name, as synonym of *referent* (a formal definition of the term *referent* will be given afterward).

The phenomenon of *Name Ambiguity* has been simplified by Kripke (1972)

as follows: “For language as we have it, we could speak of names as having a unique referent if we adopted a terminology, analogous to the practice of calling homonyms distinct 'words', according to which uses of *phonetically same sounds* to name *distinct objects* count as *distinct names*”. In his three lectures about “Naming and Necessity” Kripke (1972) uses the following formalization, that will be exploited in the remaining of this thesis:

Name A proper name, i.e., the name of a person, a city, a country, etc., excluding *definite descriptions*, but only considering those things which in ordinary language would be called proper names.

Definite Description Phrases of the form *the x such that* $f(x)$, such as 'the man who corrupted Hadleyburg'.

Designator A common term to cover names and descriptions.

Referent The object uniquely satisfying the conditions in the definite description. If exists a description of the form *the x such that* $f(x)$, and there is exactly one x such that $f(x)$, that is the referent of the description.

Identity Statement The way to express that two names have the same referent. For example 'Hesperus is Phosphorus' expresses that you see a star in the evening and it is called 'Hesperus', a star in the morning and it is called 'Phosphorus'. Then, it is found that it is not a star, but the planet Venus and that Hesperus and Phosphorus are in fact the same.

It must be specified, as introduced by Searle (1958), that the referent of a name can be determined not by a single description but by a cluster or family of descriptions. It is whatever in some sense satisfies enough or most of the family of descriptions.

This thesis focuses on the importance of Proper Names within the language, and consequently within the text, and takes into account the techniques that involve the management of Named Entities. The main goal of the thesis' work is to advance the state of the art in research on Named Entity Disambiguation.

The discussion will take into account several topics. It will concentrate on the role of Named Entities in the field of Natural language Processing and Information Extraction. In particular the tasks of Named Entity Recognition and Named Entity Disambiguation will be addressed, with focus on the second one. It will be discussed how Named Entities can be helpful for **Semantic Representation of Text**. The thesis will also present **Named Entity Disambiguation Strategy** and will investigate on what is the best strategy for NED, by means of **Empirical Evaluation**. The use of NEs will be also viewed within real **Applications**.

The contributions of this research could be summarized as follows:

- A knowledge-based strategy for NED, using Wikipedia, is presented;
- A Semantic Relatedness strategy for NED, using Wikipedia, is presented;
- As a proof-of-concept, two applications for the taking advantage of NEs are presented: a Semantic Search Engine that tries to capture document semantic using different levels of representation and an Electronic Performance Support System.

The remaining of the thesis is structured as follows:

- **Chapter 2** – *Named Entities* – gives an overview of previous work in the area of NER and NED, drawing a large variety of techniques developed using different strategies (knowledge-based, supervised learning and unsupervised).
- **Chapter 3** – *Named Entity Disambiguation Algorithms* – proposes two methods to solve the problem of name ambiguity: $WibNED$ and $SRaNED$ and presents a NLP framework that encapsulate several basic text processing techniques, together with techniques for Word Sense Disambiguation and NED.
- **Chapter 4** – *Evaluation* – provides a description of datasets and methods used for the evaluation of NED systems and reports the results of the evaluation adopting an "in vitro" strategy.

- **Chapter 5** – *Applications* – this chapter proposes two real applications, an Information Retrieval System and Electronic Performance Support System, exploiting Named Entities.
- **Chapter 6** – *Conclusions and Future Work* – concludes the thesis by summarizing ideas and results and by depicting possible future directions for the presented research work.

Chapter 2

Named Entities

2.1 Named Entity Recognition

The term "Named Entity" has been formally defined, in the field of Computer Science, during the Sixth Message Understanding Conference (MUC-6) [Grishman and Sundheim (1996)], organized by DARPA. Named Entities are those expressions that refer to people, places, organizations, products, companies, and even dates, times, or monetary amounts. Named Entity Recognition involves the identification and classification of named entities.

The big frame that encompasses the interest on Named Entites is Information Extraction (IE), that was the focus of Message Understanding Conference (MUC) series: deriving structured data from unstructured text. Named Entities played an important role for such a big issue, that is why, during MUC-6, Named Entity Recognition (NER) was defined as a sub-task of IE. Actually some research work has been carried out even before the task gained status and visibility. An example of early work on a kind of NER task is a rule-based system proposed by Rau (1991) which extracts and recognizes company names.

2.1.1 Named Entity Recognition Evaluation Contests

After MUC-6 in 1996 there has been a significative number of events dedicated to the NER task. Multilingual benchmarking and evaluations have been performed within several events, such as the International Confer-

ence on Language Resources and Evaluation (LREC), the Computational Natural Language Learning (CoNLL) workshops [Tjong Kim Sang (2002); Tjong Kim Sang and De Meulder (2003)], the Automatic Content Extraction (ACE) series organized by NIST [Doddington et al. (2004)], the Multilingual Entity Task Conference (MET), the Information Retrieval and Extraction Exercise (IREX) [Sekine and Isahara (2000)].

Also non-english communities demonstrated interest for NER: evaluation for Italian language has been performed in the context of Evalita[1], using part of the Italian Content Annotation Bank (I-CAB)[2] as evaluating corpus. For Portuguese language the competition HAREM [Santos et al. (2006)] has been organized, but also the task gained great interest within Chinese and Japanese communities. In the following, an excursus of such evaluation contests will be offered.

MUC

The Message Understanding Conferences (MUC) were initiated and financed by DARPA to encourage the development of new and better methods of information extraction, by organizing competitions requiring the development of standards for evaluation. At the sixth conference (MUC-6) the task of recognition of named entities and coreference was added and it was also re-proposed at MUC-7. For named entities were intended all phrases in the text supposed to be marked as person, location, organization, time or quantity. The Named Entity task consists of three subtasks (entity names, temporal expressions, number expressions). The expressions to be annotated are "unique identifiers" of entities (organizations, persons, locations), times (dates, times), and quantities (monetary values, percentages). The task is to identify all instances of the three types of expressions in each text in the test set and to subcategorize the expressions. The three subtasks correspond to three SGML (Standard Generalized Markup Language) tag elements: ENAMEX, TIMEX, and NUMEX. The subcategorization is captured by a SGML tag attribute called TYPE, which is defined to have a different set of possible values for each tag element.

[1] http://evalita.itc.it/

[2] http://tcc.itc.it/projects/ontotext/i-cab/download-icab.html

The output of the systems to be evaluated is in the form of SGML text markup. The only insertions allowed during tagging are tags enclosed in angled brackets. The markup has the following form:

```
<ELEMENT-NAME ATTR-NAME="ATTR-VALUE" ...>
text-string
</ELEMENT-NAME>
```

Example:

```
<ENAMEX TYPE="ORGANIZATION">Taga Co.</ENAMEX>
```

Named Entities are marked with the ENAMEX tag element. The ENAMEX subtask is limited to proper names, acronyms, and perhaps miscellaneous other unique identifiers, which are categorized via the TYPE attribute as follows:

ORGANIZATION named corporate, governmental, or other organizational entity

PERSON named person or family

LOCATION name of politically or geographically defined location (cities, provinces, countries, international regions, bodies of water, mountains, etc.)

Temporal Expressions are marked with the TIMEX tag element. The tagged tokens are categorized only via the TYPE attribute as follows:

DATE complete or partial date expression

TIME complete or partial expression of time of day

Number Expressions are marked with the NUMEX tag element. This subtask is for two useful types of numeric expressions, monetary expressions and percentages. The numbers may be expressed in either numeric or alphabetic form.
The task covers the complete expression, which is categorized via the TYPE attribute as follows:

MONEY monetary expression

PERCENT percentage

An example of text from MUC-7 (New York Times News Service) in English follows.

```
The <ENAMEX TYPE="LOCATION">U.K.</ENAMEX>
satellite television broadcaster said its subscriber base
grew <NUMEX TYPE="PERCENT">17.5 percent</NUMEX>
during <TIMEX TYPE="DATE">the past year</TIMEX>
to 5.35 million
```

The general method employed to analyze the MUC-6 results is the Approximate Randomization method, a computer intensive method which approximates the entire sample space in such a way as to allow us to determine the significance of the differences in F-Measures between each pair of systems and the confidence in that significance. The parameters in the F-Measure used are such that recall and precision scores are combined with equal weighting. The Named Entity task has been re-proposed at MUC-7 [Chinchor (1998)], the last in the series of Message Understanding Conference Evaluations, and concurrently with the English task a specific task for Chinese and Japanese was carried out (Multilingual Entity Task Conference MET-2).

ACE

The objective of the Automatic Content Extraction (ACE) program is to develop content extraction technologies to support automatic processing of human language in text form. The program is devoted to three source types: newswire, broadcast news (with text derived from Automatic Speech Recognition), and newspaper (with text derived from Optical Characters Recognition). ACE is aimed at supporting various classification, filtering, and selection applications by extracting and representing language content (i.e., the meaning conveyed by the data). Thus the ACE program requires the development of technologies that automatically detect and characterize this meaning. The ACE program consists of several phases and the first one is Entity Detection and Tracking (EDT). The taxonomy of entities to be recognized, shown in figure 2.1, is finer-graind than MUC specifications.

Type	Subtypes
FAC (Facility)	Airport, Building-Grounds, Path, Plant, Subarea-Facility
GPE (Geo-Political Entity)	Continent, County-or-District, GPE-Cluster, Nation, Population-Center, Special, State-or-Province
LOC (Location)	Address, Boundary, Celestial, Land-Region-Natural, Region-General, Region-International, Water-Body
ORG (Organization)	Commercial, Educational, Entertainment, Government, Media, Medical-Science, Non-Governmental, Religious, Sports
PER (Person)	Group, Indeterminate, Individual
VEH (Vehicle)	Air, Land, Subarea-Vehicle, Underspecified, Water
WEA (Weapon)	Biological, Blunt, Chemical, Exploding, Nuclear, Projectile, Sharp, Shooting, Underspecified

Figure 2.1: Entity types and subtypes at ACE 2007

CoNLL

The shared task of CoNLL-2003 concerns language-independent NER. It is focused on four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. The participants of the shared task have been offered training and test data for two languages, English and German. English data include foreign person names, a wide diversity of locations (including sport venues such as The Oval, and rare location names such as Nirmal Hriday), many types of organizations (from company names, such as 3M, to acronyms for political parties, such as KDP, to location names used to refer to sport teams, such as Cleveland), and a wide variety of miscellaneous named entities (from software, such as Java, to nationalities, such as Basque, to sporting competitions, such as 1,000 Lakes Rally). For each language, additional information (lists of names and non-annotated data) have been supplied as well. The challenge for the participants is to find ways of incorporating this information in their system.

HAREM

HAREM is the first evaluation contest of NER for Portuguese, and its call for participation was announced in September 2004 and the contest happened in February 2005. HAREM stands for "HAREM é uma Avaliação de Reconhecedores de Entidades Mencionadas"(roughly "HAREM is a NER Evaluation"). HAREM's shared task was to detect, and correctly annotate, named entities in Portuguese texts, in several varieties, but most prominently including Brazilian and European Portuguese documents in several genres (newspapers, Web pages, original and translated fictions, transcribed oral interviews, etc.). The methodology used in HAREM was the following:

1. Participants annotate a subset of a large corpus with several text genres, following a set of guidelines and categories for identification and classification in HAREM's golden collection (in Portuguese).
2. The collection of all participants' subsets, after due revision, constitutes what we call the Golden Collection, i.e. a NE-annotated corpus (of which only a fraction is seen by each participant prior to the contest).

3. The text of this collection (without the annotation) is then embedded in a larger collection, the HAREM collection, which the participants receive in the real contest, to annotate and send back the result.
4. Based on the information in the Golden Collection, the organization then computes and publishes the results of each system.

EVALITA: Named Entity Recognition Task

In the NER task of EVALITA, systems is required to recognize the Named Entities occurring in a text. In particular, the task is focused on the following types of Named Entities: Person (PER), Organization (ORG), Location (LOC) and Geo-Political Entities (GPE). Participants have been provided with a development corpus annotated with entities in the IOB format, that will be further described in Section 2.1.2. Both development data (335 news stories, for a total of 113,000 words) and test data (190 news stories, for a total of 69,000 words) are part of the Italian Content Annotation Bank (I-CAB), developed in the context of the Ontotext Project [3]. All data are also annotated with Part of Speech (POS) information using the Elsnet corpus for Italian [4].

2.1.2 IOB annotation format

IOB is a tagging scheme originally designed by Ramshaw and Marcus (1995). The acronym stands for "I-inside", "O-outside" and "B-begin". "I-inside" and "B-begin" denote the tokens belonging to NE while "O-outside" is used for all other tokens. IOB files contain one word per line with empty lines representing sentence (document) boundaries. At the end of each line, a tag states whether the current word is inside a NE. Such tag also encodes the type of NE. Each line contains four fields: the word, its POS tag, its chunk tag and its named entity tag.

For the entity tag, words tagged with O are outside of NEs and the I-XXX tag is used for words inside a NE of type XXX. Whenever two entities of type XXX are next to each other, the first word of the second entity will be tagged B-XXX in order to show that it starts another entity.

[3] http://tcc.fbk.eu/projects/ontotext/index.html
[4] http://historical.ncstrl.org/tr/fulltext/tr/ercimcnrcnuce/1995-TR-005.txt

The chunk tag captures three simple categories: NP (Noun Phrase), VP (Verb Phrase) and PP (Preposition Phrase). It follows the same scheme as the entity tag, with the convention I (inside), O (outside), or B (begin). A token is tagged as B if it marks the beginning of a chunk. Subsequent tokens within the chunk are tagged I. All other tokens are tagged O. The B and I tags are suffixed with the chunk type, e.g. B-NP, I-NP. Of course, it is not necessary to specify a chunk type for tokens that appear outside a chunk, so these are just labeled O.

An example fragment of IOB file follows:

```
U.N.              NNP   I-NP   I-ORG
official           NN   I-NP   O
Ekeus             NNP   I-NP   I-PER
heads             VBZ   I-VP   O
for                IN   I-PP   O
Baghdad           NNP   I-NP   I-LOC
```

The IOB-2 format, differently from the IOB one only contains two fields per line: the word and the entity tag, POS tag and chunk tag are not present.

2.1.3 NER Techniques

In more than one decade, a lot of work has been carried out in the field of NER with different and innovative methodologies. Approaches to the NER task can be roughly classified as: **deductive**, relying on heuristic rules (e.g. regular expressions), or **inductive**, using machine learning to build recognition rules, instead of completely manually defining them. Regardless of the adopted approach, NER can take advantage of external resources, such as dictionaries, lexical resources, encyclopedias and so on, both to build recognition rules and to train learning methods with additional features. A description of the main techniques follows, while an illustration of useful external resources will be presented in Section 2.3.

Rule-based Methods

Historically, handcrafted rules were the first choice to design a solution to NER [Grishman and Sundheim (1996)], and when training examples are not available, handcrafted rules remain the preferred technique, as shown recently [Satoshi Sekine (2004)]: the authors used this technique to develop a NER system for 200 entity types (the taxonomy of entities is shown in figure 2.2).

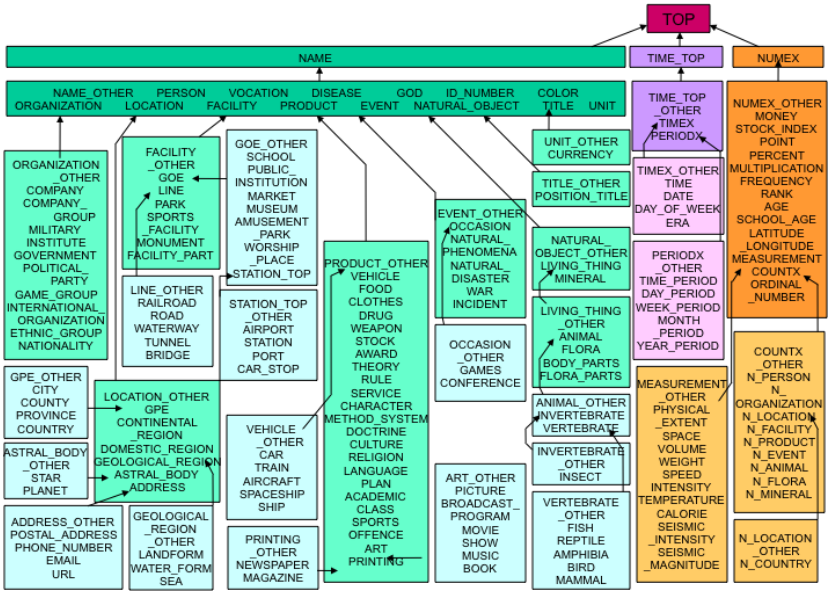

Figure 2.2: A taxonomy of around 200 entity types proposed by Sekine and Nobata (2004)

Rule-based approaches typically use manually constructed finite state patterns, which attempt to match against a sequence of words, in the same way as a general regular expression matcher. Typical systems, mainly rule-based, are University of Sheffield's LaSIE-II [Humphreys et al. (1998)], ISOQuest's NetOwl [Krupka and Hausman (1998)] and University of Edinburgh's LTG [Mikheev et al. (1998, 1999)] for English NER. However, rule-based approaches lack the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules to maintain optimal performance and the maintenance costs could be quite steep.

Learning Methods

Handcrafted rules and recognition patterns, even when accurately designed, can fail at recognizing previously unknown entities. The challenge is to automatically induce rule-based systems or sequence labeling algorithms starting from a collection of training examples. Generally speaking, different learning approaches are suitable to cope with the NER task. These can be characterized as supervised, semi-supervised or unsupervised. Supervised Learning (SL) is based on the idea of capturing the features of positive and negative examples of NEs over a large collection of annotated documents and design rules that model instances of a given type. This kind of approach is feasible when a large annotated corpus is available. Without this condition it is to be considered that the cost of creating such resources from scratch can be unaffordable. That is why alternative learning methods should be considered. In particular semi-supervised learning (SSL) (also called weakly supervised) and unsupervised learning (UL).

Supervised Learning SL techniques include Hidden Markov Models (HMM), Decision Trees (DT), Maximum Entropy Models (ME), Support Vector Machines (SVM) and Conditional Random Fields (CRF). These are all variants of the SL approach that typically consists of a system that read a large annotated corpus, memorize lists of entities, and creates rules based on discriminative features.

Hidden Markov Models Hidden Markov Models (HMMs) [Rabiner (1990)] are a type of directed graphical model.

Formally a HMM can be defined as a tuple: $< \mathbf{S}, \mathbf{W}, \pi, \mathbf{A}, \mathbf{B} >$, where

- $\mathbf{S} = \{\mathbf{s_1}, ..., \mathbf{s_n}\}$ is the set of states in the model;
- $\mathbf{W} = \{\mathbf{w_1}, ..., \mathbf{w_n}\}$ is the set of observations or output symbols;
- π are a-priori likelihoods, that is the initial state distribution:
  $\pi = \{\pi_\mathbf{i}\}$, $\mathbf{i} \epsilon \mathbf{S}$;
- A $= \{\mathbf{a_{ij}}\}, \mathbf{i}, \mathbf{j} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}\}$, is a matrix describing the state transition probability distribution:
  $\mathbf{a_{ij}} = \mathbf{P}(\mathbf{X_{t+1}} = \mathbf{s_j} | \mathbf{X_t} = \mathbf{s_i})$;

- $\mathbf{B} = \{\mathbf{B_{ij_k}}\}, \mathbf{i}, \mathbf{j} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}\}, \mathbf{k} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{t}\}$, is a matrix describing the observation symbol probability distribution:

  $\mathbf{b_{ij_k}} = \mathbf{P}(\mathbf{O_t} = \mathbf{w_k} | \mathbf{X_t} = \mathbf{s_i}, \mathbf{X_{t+1}} = \mathbf{s_j})$;

A HMM is in substance a finite state automaton with a probabilistic flavor. The model works on a graph where nodes are hidden states and edges represent transitions between them. For each state there is a probability distribution over all possible transitions and each output symbol has an emission distribution. A HMM produces a string of symbols with a generative process: given a starting state it iteratively moves to another state according to the corresponding transition probability. This process of transitioning from one state to another continues until it reaches another distinguished state, the final state. Each time a transition is made from a state, an output symbol is generated according to the symbol emission probability distribution of that state. In a typical application of HMMs, a sequence of symbols is given together with an HMM that is assumed to have produced it, the goal is to find the sequence of states that is most likely to have generated the given sequence of symbols. For example, in the case of NER, each state corresponds to a NE inside tag or the outside tag, whereas symbols correspond to words. Given a particular sentence, the NEs are defined by the most likely sequence of states that generated that sentence.
One famous example of HMM for NER is the *IdentiFinder* algorithm, within the Nymble system [Bikel et al. (1997)], another is the HMM-based chunk tagger by Zhou and Su (2002).

HMMs have been successfully used for speech recognition, as well as other natural language tasks, such as POS tagging. What plays in favour of HMMs is that the inference problem can be solved in time linear with the number of observed symbols using dynamic programming [Rabiner (1990)]: specifically the Viterbi algorithm [Forney (1973)] is used in decoding the NE-class state sequence.
An example of usage of HMM, given by Rich (1972), is reported in Appendix A, while the Viterbi algorithm is presented in Appendix B.

Decision Trees A decision tree consists of three basic elements: future, history and questions. Future consists of the possible outputs of

the model. History is the information available to the model (in the case of NER, words in the text). Questions define the decision tree: a good decision tree is determined by a good sequence of questions, considering that the choice of the m^{th} question to ask is determined by the answers to the previous $m-1$ questions. Given a specific task, the challenge is to find the better way to build a decision tree to give a solution to such task. The basic idea is to build a tree that at each point asks the question that reduces uncertainty about the set of futures. An example of DT for the NER task is the New York University's system [Sekine (1998)] . Sekine used a well known state of the art technique proposed by Quinlan (1993), the C4.5 algorithm, to grow his NER tagger decision tree. The basic idea behind C4.5 is to choose the questions that lead to greatest reduction in conditional entropy. The example used in [Sekine (1998)] is reported in Appendix C.

Maximum Entropy Models The principle of Maximum Entropy was first expounded by Jaynes (1957). It is a postulate which states that, subject to known constraints (called testable information), the probability distribution which best represents the current state of knowledge is the one with largest entropy.

Maximum Entropy Modeling has been successfully applied to many fields, including Natural Language Processing, with first examples appearing in the nineties [Berger et al. (1996); Pietra et al. (1997)]. Borthwick (1999) introduced the Maximum Entropy Models (ME) for the task of NER. He describes ME drawing on decision trees models. In this guise, ME is a method of statistical learning composed by three basic elements: futures, histories and features. Futures are possible outputs of the model. Histories are all of the conditioning data which permit to assign probabilities to the space of futures. Features are binary functions of the history and future. An example of a feature denoted by g follows:

$$g(h,f) = \begin{cases} 0 & \text{if } current\_token\_capitalized(h) \text{ and } f = location\_start; \\ 1 & otherwise \end{cases}$$

In the example above the *history* h is the token whose tag is to be decided. The predicate $current\_token\_capitalized(h)$ states that h starts

with a capital letter. The feature g determines that in this case the word is more likely to be the start of a location name.

The computation of $P(f|h)$ in ME is dependent on a set of features which, hopefully, are helpful in making a prediction about the future. Given a set of features and some training data, the process of ME estimation produces a model in which each feature g_i has an associated parameter α_i, called weight. The conditional probability of the future, given the history, is the product of the weights for all features and it is computed as follows:

$$P(f|h) = \frac{\prod_i \alpha_i^{g_i(h,f)}}{Z_\alpha(h)} \tag{2.1}$$

where Z is a normalization function

$$Z_\alpha(h) = \sum_f \prod_i \alpha_i^{g_i(h,f)} \tag{2.2}$$

Chieu and Ng (2003) adopt the maximum entropy approach to classify words in a document D as one of the following:

- the beginning of a NE (B tag)
- a word inside a NE (C tag)
- the last word of a NE (L tag)
- the unique word in a NE (U tag)

In their method it is possible that the classifier produces a sequence of inadmissible classes (e.g., PER-B followed by LOC-L). To eliminate such sequences, the authors define a transition probability between word classes $P(c_i|c_j)$ to be equal to 1 if the sequence is admissible, and 0 otherwise. The probability of the classes $c_1, \ldots, c_n$ assigned to the words in a sentence s in a document D is defined as follows:

$$P(c_1, \ldots, c_n|s, D) = \prod_{i=1}^{n} P(c_i|s, D) * P(c_i|c_{i-1})$$

where $P(c_i|s, D)$ is determined by the maximum entropy classifier. The Viterbi algorithm is then used to select the sequence of word classes with the highest probability.

Support Vector Machines Support Vector Machines (SVM) have been introduced in the nineties [Vapnik (1995)]. SVMs are a set of supervised learning methods used for classification and regression. In a binary classification setting, given a set of training examples, each marked as belonging to one of two categories, a training algorithm builds a model that predicts the category of any new example. Intuitively, a SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. A good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Ideally an SVM analysis should produce a hyperplane that completely separates the feature vectors into two non-overlapping groups. However, perfect separation may not be possible, or it may result in a model with so many feature vector dimensions that the model does not generalize well to other data; this is known as over fitting. To allow some flexibility in separating the categories, SVM models have a cost parameter, c, that controls the trade off between allowing training errors and forcing rigid margins. It creates a soft margin that permits some misclassifications. Increasing the value of c increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well. Given some training data, a set of n points of the form

$$D = \{(x_i, c_i) \mid x_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the c_i is either 1 or -1, indicating the class to which the point x_i belongs. Each x_i is a p-dimensional real vector. We want to find the maximum-margin hyperplane which divides the points having $c_i = 1$ from those having $c_i = -1$. Any hyperplane can be written as the set of points X satisfying $w \cdot x - b = 0$, where $\cdot$ denotes the dot product. The vector w is a normal vector: it is perpendicular to the hyperplane. The parameter

$\frac{b}{||w||}$ determines the offset of the hyperplane from the origin along the normal vector w.

We want to choose the w and b to maximize the margin, or distance between the parallel hyperplanes that are as far apart as possible while still separating the data. These hyperplanes can be described by the equations $w \cdot x - b = 1$ and $w \cdot x - b = -1$.

Yamcha [Kudo and Matsumoto (2003)] is a popular and freely available NER annotator which uses the SVM technique. It has been employed within the thesis work to implement the task of NER.

Conditional Random Fields Conditional Random Fields (CRF) is a framework for building probabilistic models to segment and label sequence data. CRFs are a type of undirected graphical model in which each vertex represents a random variable whose distribution is to be inferred, and each edge represents a dependency between two random variables. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states with few successor states [Lafferty et al. (2001)]. X is a random variable over data sequences to be labeled, and Y is a random variable over corresponding label sequences. All components Y_i of Y are assumed to range over a finite label alphabet Y . For example, X might range over natural language sentences and Y range over part-of-speech taggings of those sentences, with Y the set of possible part-of-speech tags.

Definition 1 *Let $G = (V, E)$ be a graph such that $Y = (Y_v)_{v \in V}$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field in case, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: $p(Y_v \mid X, Y_w, w \neq v) = p(Y_v \mid X, Y_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.* [Lafferty et al. (2001)]

CRFs have been largely used as a solution for NER. McCallum and Li (2003) present results on the CoNLL-2003 named entity recognition

(NER) shared task. Their method obtains overall English F-score of 84.04% on the test set by using CRFs, feature induction and Web-augmented lexicons. Sarawagi and Cohen (2004) compared CRF algorithms on NER problems with three different corpora: they extracted city names and state names from the Address corpus (395 home addresses of students in a major university in India [Borkar et al. (2001)]), company names and job titles from the Jobs corpus [Califf and Mooney (2003)], consisting of 300 job postings, and person names from the CSPACE email corpus [Minkov et al. (2005)].
Okanohara et al. (2006) propose a framework of semi-CRF model for NER, introducing the employing of a filtering process to remove unnecessary state candidates beforehand and then using the feature forest model [Yusuke and Jun'ichi (2002)], which enables to pack feature-equivalent states. Experimental results show that the proposed model achieves an F-score of 71.48% on the JNLPBA 2004 shared task [Kim et al. (2004)] without using any external resources or post-processing techniques. CRFs have also achieved good results also in the bio-NER task [Settles (2004)].
3

Weakly Supervised Learning The major technique in the category of weakly supervised methods is called *bootstrapping*. In this method, a small degree of supervision, such as a set of seeds, is used at the beginning.

There are several examples of this technique. Nadeau et al. (2006) presents a system able to extract "disease names", given a few initial disease. Then the system searches the sentences that contain the names, and finds the strong indicators of context where the five disease names appear. Then, the system tries to find other instances that appear in the context. By repeating these processes, a large number of disease names can be gathered.
Other examples of this type of research include [Riloff and Jones (1999); Yangarber et al. (2002); Collins and Singer (1999); Brin (1998); Lin (2000)].

Unsupervised Learning The typical approach of unsupervised learning is clustering. For example, we can try to gather named entities from the clustered groups based on the similarity of context.

In the field of NER have been carried out several research works exploiting this technique. The approach proposed by Alfonseca and Manandhar (2002) exploits WordNet as NE types provider (e.g., location>country, animate>person, animate>animal, etc.), assigning a topic signature to each WordNet synset by merely listing words that frequently co-occur with it in a large corpus. Then, given an input word in a given document, the word context (words appearing in a fixed-size window around the input word) is compared to type signatures and classified under the most similar one.

Shinyama and Sekine (2004) used an observation that named entities often appear synchronously in several news articles, whereas common nouns do not. They found a strong correlation between being a named entity and appearing punctually (in time) and simultaneously in multiple news sources. This technique allows identifying rare named entities in an unsupervised manner and can be useful in combination with other NERC methods.

In [Etzioni et al. (2005)], Pointwise Mutual Information and Information Retrieval (PMI-IR) is used as a feature to assess that a named entity can be classified under a given type. PMI-IR [Turney (2001)], measures the dependence between two expressions using web queries. A high PMI-IR means that expressions tend to co-occur.

2.1.4 Named Entity Recognition Features

Given a text, each word can be labeled with a set of descriptors. Information gathered by these attributes, named features, enrich the text with meta information intended for algorithmic use. An example of a feature is a Boolean variable with the value true if a word is capitalized and false otherwise. NER problem is usually approached by applying a rule system over the features. To give a toy example, we could recognize all candidate entities with the rule "capitalized words are candidate entities" and classify them as organization if "the length of the candidate entity is greater than 3 words". Nadeau and Sekine (2007) provide a survey on NER and report examples of features, proposed in table 2.1.

Digits can represent dates, percentages, intervals, identifiers, etc. Some work focused the attention on particular patterns of digits: two-digit and

Table 2.1: Examples of features

Features	Examples
Case	Starts with a capital letter
	Word is all uppercased
	The word is mixed case (e.g., ProSys, eBay)
Punctuation	Ends with period, has internal period (e.g., St., I.B.M.)
	Internal apostrophe, hyphen or ampersand (e.g., O'Connor)
Digit	Digit pattern
	Cardinal and Ordinal
	Roman number
	Word with digits (e.g., W3C, 3M)
Character	Possessive mark, first person pronoun
	Greek letters
Morphology	Prefix, suffix, singular version, stem
	Common ending
Part-of-speech	proper name, verb, noun, foreign word
Function	Alpha, non-alpha, n-gram
	lowercase, uppercase version
	pattern, summarized pattern
	token length, phrase length

four-digit numbers can represent years [Bikel et al. (1997)] and when followed by a "s" they can stand for a decade; one and two digits may stand for a day or a month [Wu (1998)].

Morphological features have been previously exploited for the NER task. Suffixes turned out to be useful in different cases: a human profession for example often ends in "ist", e.g. journalist, cyclist; nationality and languages often end in "ish" and "an", e.g. Spanish, Danish, Romanian; organization names often end in "ex", "tech", and "soft" [Bick (2004)].

Applying functions over words is another way of constructing features. Examples are provided in [Collins and Singer (1999)], in which the authors isolated the non-alphabetic characters of a word, and in [Patrick et al. (2002)] where character n-grams have been used as features. Other types of functions over words are pattern features that map words onto a small set of patterns over character types [Collins (2002); Cohen and Sarawagi (2004); Settles (2004)]. For instance, a pattern feature might map all uppercase letters to "A", all lowercase letters to "a", all digits to "0" and all punctuation to "-".

Other features can be gathered from the occurrence of words in gazetteer (lists of cities, proper names, organizations), or in lookup lists. More sophisticated features can be used too, such as the output of other named entity classifiers.

2.2 Named Entity Disambiguation

NER, as described in Section 2.1, is aimed at finding occurrences within a text of certain types of entities, where the entity types are domain dependent. NER produces a many-to-many mapping between names and entities, due to two well known phenomena in Natural Language, that is synonymy and polysemy, that in the case of NE are better defined by the concepts *variation* and *ambiguity*. Named Entity Disambiguation (NED) is the problem of mapping mentions of entities in a text with the object they are referencing. It is a step further from NER and associates names with entities that are predefined in an external repository, what we call a *name inventory*: the task establishes a unique mapping between a mention of name in a text, the *surface form*, and the real world object in

the *name inventory*. It can be assumed to have a dictionary of all possible entity entries. The NED process aims to create a mapping between the *surface form* of an entity and its unique dictionary meaning.

Different definitions for the task can be found in different works. Here some of them are reported.

Definition 2 *Entity disambiguation resolves the many-to-many correspondence between mentions of entities in text and unique real-world object.* [Blume (2005)]

Definition 3 *Entity disambiguation (sometimes also referred to as entity tracking) is the process of determining which names, words, or phrases in text correspond to distinct persons, organizations, locations, or other entities.* [Blume (2005)]

Definition 4 *Named Entity Disambiguation or Proper Name Disambiguation is a type of Word Sense Disambiguation in which words to be disambiguated are Named Entities.* [García et al. (2007)]

Definition 5 *Entity Disambiguation is the challenge of determining the correct entity out of various candidate entities.* [Hassell et al. (2006)]

Definition 6 *Entity Disambiguation is the problem of distinguishing the real world referent of a given name in context.* [Mann and Yarowsky (2003)]

Definition 7 *An entity is defined as an element from a population of objects. Rather, there exist a set of objects which are used to reference entities and take the form of names. The set of distinct names N_i observed in a text is represented by $E = \{e_1, e_2, \ldots, e_n\} = N_1 \cup N_2 \cdots \cup N_m$. While the same name can be ambiguous to multiple entities, each occurrence of a name references a single ambiguous names. A name which refers to k different entities is called k-ambiguous.* [Malin (2005)]

Definition 8 *Named Entitiy Disambiguation aims at locating in a Knowledge Base of discourse the entity that a name represents.* [Nguyen and Cao (2008)]

Definition 9 *The reference disambiguation corresponds to ensuring that references in a database point to the correct entities. In the reference*

disambiguation problem, for each reference, a set of possible candidates is given and the task is to pinpoint the correct entity in this set. [Nuray-Turan et al. (2007)]

The task of Named Entity Disambiguation, with different nuances, it is addressed with other definitions, such as Entity Resolution, Coreference Resolution, Entity Discrimination, but all of them have different specific goal and settings.

Indeed, **Entity Resolution** is the problem of determining which records in a database refer to the same entities. It was first introduced by Newcombe et al. (1959). Most current approaches are variants of the Fellegi-Sunter model [Fellegi and Sunter (1969)], the first to give a statistical formulation to the problem, viewed as a classification problem: given a vector of similarity scores between the attributes of two entities, classify it as "match"or "non-match".

Coreference Resolution is the task which aims to link together name variants of the same entity appearing in a text.

The task of **Named Entity Discrimination** concerns clustering multiple occurrences of a name into classes. Entity Discrimination does not rely on an external repository for sense definitions and in this sense is cheaper in terms of resources, because no external source of knowledge is needed. Also, may be considered as easier than disambiguation, because there is no request for this task to identify the actual entity referred from a textual evidence, but only if two or more mentions of names refer to the same entity. When extended to general noun phrases, the discrimination problem is also known as Coreference Resolution.

Named Entity Disambiguation, as intended in this dissertation, is the task of mapping the list of entities appearing in a text with the correct unique real-world objects respectively referred. The occurrence of an entity in the text with will be denominate with the term *reference* and the addressed real world object with the term *referent*. It would be assumed to have an *entity inventory*, containing all possible referents, and that each reference has associated the set of candidate entities, that is all real-world objects that a single reference could refer. Hereafter will be explored the variety of methodologies employed to solve the NED task.

2.2.1 Knowledge-based Methods

A milestone study, declaratively addressing the problem of Named Entity Disambiguation, is proposed by Wacholder et al. [Wacholder et al. (1997)] and presents the Nominator system, developed at IBM. The proposed model focuses on two specific problems: the recognition of proper names within an unstructured text and the discovery of new names. The authors point on the expensiveness of models that are strongly centered on computational resources or human-builded resources (such as manually annotated corpora): they propose a method which relies on little computational resources, in the form of an authority file, which contains 3,000 special names words (personal titles, organization identifiers, large places, exception capitalized words, such as TV, I, Very...) and 20,000 person first names. The proposed solution makes use of the above authority file together with rules, patterns and indicators within the text. Indeed the authors identify different types of ambiguity in a name:

- *structural ambiguity*: within Prepositional Phrases and in Conjunctions it is difficult to correctly identify the boundaries of a proper name (e.g. The Museum of Modern Art in New York City)
- *semantic ambiguity*: the same name can refer to different objects (e.g. Ford can be a person, an organization, a make of car, a place...)
- *proper name/common name*: lots of common names can also be proper names (e.g. house/The House, candy/Candy...) but also naming conventions can be often disregarded, and a woman can be named April Wednesday.

After a first step of Name Discovery, which consists in splitting the text in tokens and generating a candidate name list, several steps are performed, with respect to different kind of ambiguity. For structural ambiguity some intrinsically ambiguous operators are taken into account, such as prepositions, conjunctions and possessive pronouns. For semantic ambiguity indicators (Mr., Dr. ...) and anchors (names that unambiguously refer to certain entity types) are strongly used.

In fact, the task addressed by this study is not NED as intended in this dissertation, and as it has been later defined, because there is no *name inventory*, but it is a step further from simple Named Entity Recognition because it deeply deals with proper name ambiguity.

Peng et al. (2006) proposed a knowledge based approach of processing documents to disambiguate not all types of Named Entities but only those representing locations. Lacking of training data is a problem in disambiguation method. Their method automatically extract training data from large collection of documents based on local context disambiguation, and then sense profiles based on global context are generated automatically for disambiguation use. Local context of a location mention is direct neighbor words of the mention in a document. Global context of a sense is frequently co-located words of the sense in a collection of documents. The hypothesis behind this work is that every sense of location entity has different global context, so building a profile based on this kind of information could be useful to disambiguate location entity, by looking for similar context in profiles. To build these profiles the authors generate some training data from English newspaper articles (TDT4 collection[5]), automatically disambiguating a small portion of location mentions, extracted with an Entity Recognition Tool, matching local context and information from a world gazetteer[6]: if the context of an entity appears in the parent or child node of a sense in the gazetteer, than that entity is stored as training data. All acquired training data has then been used to generate profiles, which have been indexed with a search engine tool. Disambiguation is then performed by querying the search engine over the profiles with the ambiguous location. The answer to the query is obtained linearly combining three different scores: ranking position, local context, and the popularity of individual location sense. Results over 300 articles from the collection, manually annotated to construct a ground truth, show that weighting different scores could conduct to better results.

Aswani et al. (2006) propose an instance unification methodology, that is determining whether two instances to the same object in the real world. They focus on person names within an ontology containing publications, titles, authors, abstracts, etc., where different instances of these are created from bibliography records. The ontology population has been performed automatically, assuming that all authors of all publications are different and a corresponding instance is created in the ontology for each of them. Then the addressed instance unification task is to determine

[5]http://projects.ldc.upenn.edu/TDT4/
[6]http://www.world-gazetteer.com/

the number of distinct authors and insert the required "sameIndividualAs" statements in the ontology. The proposed approach combines the use of citation information (abstract, initials, titles and co-authorship information) with web mining and charge the attention at identifying which features lead to the best performance on the author disambiguation task. Results show that the information mined from the web contributes substantially towards the successful handling of highly ambiguous cases which lowered the performance of previous methods.

Ontology-based Methods

Several works proposed NED methods where the external resource used as a *name inventory* is in the form of an ontology. García et al. (2007) proposed a method for NED in the news domain that use the NEWS ontology as *name inventory*. The proposed method, named IdentityRank, is inspired by PageRank algorithm [Brin and Page (1998)]. The basic idea of the work is the *sematic coherence* of entities, that is instances of a certain type usually occur in news of a certain category. Also the occurrence of an entity in a text gives information about other entities: similarly to what PageRank does with web pages, IdentityRank finds and ranks entities within a text, assuming that an instance has high rank if the sum of the ranks in the news item of the instances that typically co-occur with it is high. The evaluation is done using a self-builded corpus containing two ambiguous entities (Alonso, Georgia) with two different metrics: global and relative accuracy. Global accuracy is measured as total number of right assignment entity/intance of the ontology divided by the total number of assignments. Relative accuracy for the entity used to build the corpus concerns the total number of right assignment on the decisions of that entity divided by the total number of decisions of that entity. Results shown are mostly above the theoretically builded baseline.

Hassell et al. (2006) proposed a novel method for NED which utilizes background knowledge in the form of a populated ontology. The method works on unstructured text and it uses different relationships in a document as well as from the ontology to provide clues in determining the correct entity. Successful experiments of the method have been carried out

on a collection of DBWorld[7] posts using a large scale, real-world ontology extracted from the DBLP bibliography website[8]. The authors argue that rich semantic metadata representations allow a variety of ways to describe a resource. The first step of the approach is specifying which entities from a populated ontology are to be spotted in text and later disambiguated. To do this,the authors indicate which literal property is the one that contains the "name" of entities to be spotted. After spotting entity names in text every potential match with the ontology is given a confidence score. The confidence score for each ambiguous entity is then adjusted based on whether existing information of the entity from the ontology matches accordingly to the relationship types found in the ontology (such as text-proximity, text co-occurance, semantic relationsships). Evaluation has been carried out on a manually constructed dataset, consisting of 20 documents from DBLP. Each entity appearing within documents has been labelled with the corresponding DBLP author's page: this link has been used within the ontology as the URI that uniquely identifies a researcher. Results are calculated in terms of precision and recall. Given A the set of unique names from the dataset and B the set of entities identified by the proposed algorithm, then precision is the proportion of correctly identified entities with regard to B and recall is the proportion of correctly disambiguated entities with regard to A. The method reported a precision of 97.1% and a recall of 79.4%.

Nguyen and Cao (2008) present a method that overcomes the problem of the shortage of available training data, by automatically generating an annotated corpus based on a specific ontology. They employ a machine learning model to disambiguate and identify named entities, by using an unsupervised method, based on Harris' Distributional Hypothesis [Harris (1954)], stating that words occurring in similar contexts tend to have similar senses. The proposed method also aims at exploring meaningful features for representing NEs in texts and a Knowledge Base (KB), then assigning each NE referred to in a text to the contextually most similar instance in the KB. Empirical evaluation shows that, while the ontology provides basic features of named entities, Wikipedia is a fertile source for additional features to construct accurate and robust named entity

[7] http://www.cs.wisc.edu/dbworld/

[8] http://www.informatik.uni-trier.de/ ley/db/

disambiguation systems.

Wikipedia-based Methods

Many studies that exploit Wikipedia as a knowledge source have recently emerged [Ponzetto and Strube (2006); Strube and Ponzetto (2006); Zesch et al. (2007)]. In particular, Wikipedia turned to be very useful for the problem of Named Entities due to its greater coverage than other popular resources, such as WordNet [Leacock and Chodorow (1998)] that, resembling more to a dictionary, has little coverage on named entities [Strube and Ponzetto (2006)]. Lots of previous works exploited Wikipedia for the task of NER, e.g. to extract gazetteers [Toral and Munoz (2006)] or as an external knowledge of features to use in a Conditional Random Field NER-tagger [Kazama and Torisawa (2007)], to improve entity ranking in the field of Information Retrieval [Vercoustre et al. (2008)]. On the other hand, little has been carried out on the field of NED. The most relevant works on NED based on Wikipedia are those by Bunescu and Pasca (2006) and Cucerzan (2007). Bunescu and Pasca consider the problem of NED as a ranking problem. They define a scoring function that takes into account the standard cosine similarity between words in the context of the query and words in the page content of Wikipedia entries, together with correlations between pages learned from the structure of the knowledge source (mostly using Wikipedia Categories assigned to the pages). Wikipedia has been used as a dataset of disambiguated occurrences of proper names and the context article for each entity has been taken into account within the similarity function. Their method achieved accuracy between 55.4% and 84.8% [Bunescu and Pasca (2006)].

Cucerzan proposes a very similar approach: the vectorial representation of the document is compared with the vectorial representation of the Wikipedia entities. In more details the proposed system represents each entity of Wikipedia as an *extended vector* with two principal components, corresponding to context and category information; then it builds the same kind of vector for each document. The disambiguation process consists of maximizing the *Context Agreement*, that is the overlap between the document vector for the entity to disambiguate and each possible entity vector. Cucerzan proposed the Vector Space Model as a

solutuion for the NED problem and the best result for this approach is an accuracy of 91.4% [Cucerzan (2007)].

Both presented works [Bunescu and Pasca (2006); Cucerzan (2007)] are based on the Vector Space Model, which means that a pre-computation on the Wikipedia knowledge resource is needed to build the vector representation. What is more, their methods treat content in a Wikipedia page as a bag-of-words (with the addition of categories information), without taking into account other structural elements in Wikipedia.

Han and Zhao (2009) identify the key problem of Named Entity Disambiguation in measuring the similarity between occurrences of names. Traditional methods measure the similarity using essentially two kind of models: the bag of words (BOW) or Social networks. Both kind of measures have some lititations: BOW-based measures ignore all the semantic relations such as social relatedness between named entities, associative relatedness between concepts, polysemy and synonymy between key terms. Social networks can only capture the social relatedness between named entities. To overcome these deficiencies, Han and Zhao propose to use Wikipedia as the background knowledge for disambiguation, which surpasses other knowledge bases by the coverage of concepts, rich semantic information and up-to-date content and allow to measure the similarity between occurrences of names more accurately. Given the computed similarities, name observations are disambiguated by grouping them according to their represented entities, using the hierarchical agglomerative clustering (HAC) algorithm. The proposed method has been evaluated on the disambiguation of personal name, over the standard WePS data sets [9] using WePS proposed measures: *Purity* (homogeneity of the observations of names in the same cluster), *Inverse purity* (completeness of a cluster) and *F-Measure* (harmonic mean of purity and inverse purity). Empirical results show that the disambiguation performance of our method gets 10.7% improvement over the traditional BOW based methods and 16.7% improvement over the traditional social network based methods.

[9] http://nlp.uned.es/weps/weps-1/weps-1-data

2.2.2 Learning Methods

In the ambit of Named Entity Disambiguation, Learning methods, both supervised and unsupervised, have been mostly used on a subtask of NED, which is Person Name Disambiguation.

Person Name Disambiguation

Person Name Disambiguation is a particular kind of Entity Disambiguation and focuses the attention on persons instead of entities of whatever category. Disambiguation of person names can turn to be very crucial in a Web searching scenario, has also recognized by popular press; Reuters (March 13, 2003) observed the problem of name ambiguity to be a major stumbling block in personal name web searches. The problem has also been faced in one of SEMEVAL 2007 task[10], with the goal of grouping documents referring to the same individual.

Sugiyama and Okumura (2007) propose a semi-supervised clustering approach for the task of Personal Name Disambiguation. They integrate similar documents into a labeled document. The proposed method is a step further from the pure agglomerative clustering, where initially, each Web page is an individual cluster, and then two clusters with the largest similarity are iteratively merged to generate a new cluster until this similarity is less than a predefined threshold. Authors claim that if a seed page that describes a person is introduced, the clustering for personal name disambiguation would be much more accurate. Therefore, they apply semi-supervised clustering to disambiguate personal names in Web search results, using two kinds of seed page: (a) the article on each person in Wikipedia, and (b) the top ranked Web page in the Web search results. Results show that this method, compared to the pure agglomerative clustering, generates a smaller number of clusters, making easier for a user the task of browsing Web pages clustered based on each personal entity.

Srinivasan et al. (2009) present a cross-document Person Name Disambiguation system. The goal is to cluster documents such as each cluster contains all and only those documents referring to the same person. The authors introduce entity profiles, ets of information that are col-

[10]http://nlp.uned.es/weps/

lected for each ambiguous person in the entire document. Also, they represent entities in a vector-space model (VSM), using a modified term frequency-inverse document frequency (TF-IDF) weighting scheme. Disambiguation is then performed via single-link hierarchical agglomerative clustering. Experiments are carried out on two corpora: the Bagga Baldwin corpus [Bagga and Baldwin (1998)], which contains one ambiguous name and the English Boulder name corpora, a news corpus acquired from a web search, containing four sub corpora each corresponding to one of four different ambiguous person names, already used by Chen and Martin (2007). Results show an improvement to the state of the art on same corpora, with an F-measure of 94.03%.

Han et al. (2003) take in hand the problem of author names ambiguity in publications or bibliographies: They proposed the usage of a K-means clustering algorithm based on an extensible Naive Bayes probability model. The algorithm is based on three features collected from citations: co-author names, the title of the paper and the title of the journal or proceedings. The work is based on the assumption that a researcher usually has research areas that are stable over a period and tends to co-author papers with a particular group of people during that period. The disambiguation system, given an author name, clusters the citations of different similar named entities. However, their method uses manually collected publications pages, where the correct publication pages are identified manually among the results returned by Google with a query consisting of the author name and "publication"as a keyword. The approach is evaluated on two names "J Anderson"(6) and "J Smith"(9) with accuracy of 70.6% and 73.6% respectively.

This work has been improved further in [Han et al. (2004)] by using information about aliases and name invariants from a database. Two supervised learning approaches are proposed to disambiguate authors in the citations, one based on naive Bayes probability model, the other based on Support Vector Machines (SVMs). Both approaches utilize the same three types of citation attributes as previous work: co-author names, the title of the paper, and the title of the journal or proceeding and assume the existence of a citation database (training data) indexed by the canonical name entities, i.e. a name that is the minimal invariant and complete name entity for disambiguation. Based on the observation that an au-

thor's citations usually contain the information of the author's research area and his or her individual patterns of coauthoring, the authors conjecture that a generative model like naive Bayes it is a promising choice due to the fact that it can create other examples of the data and capture all authors' writing patterns. They also investigate on the usage of a discriminative model, such as Support Vector Machines, for this task. In the SVM approach citations are represented in a vector space and each author is considered as a class: a new citation is classified to the closest author. Differently from naive Bayes SVM can learn from both positive and negative training citations. Another difference between SVM and naive Bayes is that the first classifies a citation using distance measures while the second is based on probabilities. Experimental settings have been established as follows: given a full citation with the query name implicitly omitted the goal is to predict the most likely canonical name from the citation database. Two experimental datasets have been used: one collected from the web, the other collected from the DBLP citation databases. SVM outperforms nayve Bayes, except in the case of using coauthor information alone. The reason is that SVM can better capture and rank the features unique to a class, while naive Bayes assume all features to have the same distribution.

Mann and Yarowsky (2003) present a set of algorithms for distinguishing personal names with multiple real referents in text, based on little or no supervision. The approach utilizes an unsupervised clustering technique over a rich feature space of biographic facts, which are automatically extracted via a language-independent bootstrapping process. The feature vectors for each document, have been generated using different methods, such as using all words (plain) or only Proper Nouns (nnp), using most relevant words (tf-idf), employing basic or extended biographical features. Performance is evaluated based on both a test set of hand-labeled multi-referent personal names and via automatically generated pseudonames.

Another unsupervised approach has been proposed by Chen and Martin (2007). They attack the problem of automatically separating sets of news documents generated by queries containing personal names into coherent partitions. They propose clustering as a solution, with a range of syntactic and semantic features, beyond bag of words contextual features or biographical information. In particular they extract noun phrase fea-

tures, named-entity feature, target entity, local entity (locally co-occurring entity with the target one), non-local entity. The proposed methodology follows a common architecture for named-entity disambiguation: the detection of ambiguous objects, feature extraction and representation, similarity matrix learning, and finally clustering. Evaluation over the Bagga and Baldwin corpus [Bagga and Baldwin (1998)] and against their achieved results, show an overall improved performance.

Pedersen et al. (2005) make a parallel between name discrimination, the problem of grouping occurrences of a name based on the underlying entity's identity, and word sense discrimination, the process of examining a number of sentences that contain a given polysemous word, and then grouping those instances based on the meaning of that word (differently from word sense disambiguation, the process of assigning a sense to a polysemous word from a predefined set of possibilities). The authors present an unsupervised approach that resolves name ambiguity by clustering the instances of a given name into groups, each of which is associated with a distinct underlying entity. They use statistically significant bigrams that occur in the same context as the ambiguous name as features that represent the context of the ambiguous name. They generate a high dimensional "instance by word "matrix (by manipulating bigrams) and reduce it to its most significant dimensions by Singular Value Decomposition (SVD). The different "meanings"of a name are discriminated by clustering these second order context vectors with the method of Repeated Bisections. The proposed method has been evaluated over an ad-hoc builded corpus of text containing ambiguous pseudo-names and proved to be more accurate than the majority classifier, and that the best results are obtained by having a small amount of local context to represent the instance, along with a larger amount of context for identifying features, or vice versa.

2.2.3 Graph-based methods

The problem of Entity Reference Resolution or simply the Entity Resolution problem has been addressed from a graph based data analysis point of view by Kalashnikov and Mehrotra (2005). Usually, knowledge about entities and the relationships among them resides in numerous documents and datasets distributed across a variety of data sources. Information Extraction has made possible to extract such entities and relationships automatically (at least in limited domains), which traditionally were manually collected from analysts. A key issue in constructing graphs from diverse sources is that of resolving the references to entities in these datasets. In the ideal world each entity would have a unique identifier and, whenever this entity is referred to, its unique identifier is specified so that there is no uncertainty. In the real world, however, entities are often referred to by descriptions that may be created by multiple persons and collected from various sources. Entities might be referred to differently (by different descriptions) and also multiple entities might end up matching the same description. The authors compare two different kind of approaches to such problem: Feature Based Similarity (FBS) approaches against their innovative Reference Disambiguation Approach (RDA), which adopts a probabilistic model to estimate the strength of relationships between entities in a graph.

The authors use an undirected graph to represent entities and their relations. Specifically, the graph $G(V, E)$ is composed of two sets, the set of nodes V and the set of edges E. X is the set of all entities in the dataset. Node $v_i \in V$ corresponds to entity $x_i \in X$, each edge corresponds to a relationship and is labeled with a weight. Edge weights are used to reflect the degree of confidence the relationship corresponding to the edge exists. By default all weights are equal to 1. Each entity x_i consists of a set of m_{x_i} properties $x_i.a_1, x_i.a_2, \ldots, x_i.a_{m_{x_i}}$ and contains a set of n_{x_i} references $x_i.r_1, x_i.r_2, \ldots, x_i.r_{n_{x_i}}$, where, each reference $x_i.r_k$ is semantically referring to a single entity $d(x_i.r_k)$ in X. But description provided by $x_i.r_k$ can match in general multiple entities in X. A reference $x_i.r_k$ is essentially a description and has associated with it a non-empty set of entities $CS(x_i.r_k)$, referred to as choice set for the reference $x_i.r_k$. Such a choice set consists of all the entities that $x_i.r_k$ could potentially

refer to. The authors assume that $CS(x_i.r_k)$ is given for each $x_i.r_k$ and has N (i.e., $N =\mid CS(x_i.r_k) \mid$) elements $y_1, y_2, \ldots, y_N$.

If entity y_j is chosen as the outcome of such a resolving, then $x_i.r_k$ is said to be resolved to y_j. Reference $x_i.r_k$ is said to be resolved correctly if this y_j is $d(x_i.r_k)$. Reference $x_i.r_k$ is said to be unresolved if it is not resolved yet to any y_j and $N > 1$. If $CS(x_i.r_k)$ has only one element, then $x_i.r_k$ is automatically resolved to y_1 , and graph G contains an edge between x_i and y_1. This edge is associated with a weight 1 to denote that the probability of the edge to exist is 1. If $CS(x_i.r_k)$ has more than one element, then graph G contains a choice node $Cho(x_i.r_k)$ to reflect the fact that $d(x_i.r_k)$ can be one of $y_1, y_2, \ldots, y_N$. Node $Cho(x_i.r_k)$ is linked with node $V(x_i)$ via edge $e_0 = (V(x_i), Cho(x_i.r_k))$. Node $Cho(x_i.r_k)$ is also linked with N nodes $V(y_1), V(y_2), \ldots, V(y_N)$, for each y_j in $CS(x_i.r_k)$, via edges $e_j = (Cho(x_i.r_k), V(y_j))$. Nodes $V(y_1), V(y_2), \ldots, V(y_N)$ are called the options of choice $Cho(x_i.r_k)$. Edges e_j between choice node $Cho(x_i.r_k)$ and its options $V(y_1), V(y_2), \ldots, V(y_N)$ are called the option edges of choice $Cho(x_i.r_k)$. The weigh of edge e_0 is one, weights of edges e_j for $j = 1, \ldots, N$ are undefined initially. Resolving $x_i.r_k$ means to associate weights of zero with all but one edge e_j among edges $e_1, e_2, \ldots, e_N$. To resolve an entity means to resolve all of its references. To resolve graph G means to resolve all entities in X and hence label all the option edges.

The complete entity disambiguation algorithm is called RDA (Reference Disambiguation Approach) and consists of a Probabilistic Model (PM) over the above described graph that makes use of feature-based similarity and relationship-based similarity to calculate connection strength between two entities. Experiments have been carried out on the author matching problem, which considers authors and papers: the goal is authors disambiguation. Results over three datasets (one real and two synthetic) show high disambiguation accuracy.

Nuray-Turan et al. (2007) focus the attention on calibrating a connection strength (CS) measure from training data in the context of reference disambiguation problem, that is identifying for each reference the unique entity it refers to. Given any two nodes u and v in the graph G, the connection strength c(u, v) returns how strongly u and v are interconnected to each other in G. Generally, a domain expert decides a mathematical model to compute c(u, v), which describes the underlying dataset best.

The authors propose a supervised learning algorithm that learns the importance of CS, among the classified entities and makes the approach self-tunable to any underlying domain. The algorithm uses a graphical methodology; the disambiguation decisions are made based on object features and inter-object relationships, including indirect ones that exist among objects. Experiments have been carried out on two synthetic datasets taken from two domains: Movies (from Stanford Movies Dataset) and Publications (CiteSeer dataset). Stanford Movies Dataset contains three different entity types: movies (11453 entities), studios (992 entities) and people (22121 entities) and five types of relationships: actors, directors, producers, producing studios and distributing studios. CiteSeer dataset contains four different types of entities: author, paper, department, and organization and three types of relationships: author-paper, author-department, and department-organization. The authors introduce uncertainty in both datasets manually. Results are expressed in terms of accuracy and show an increase of the quality of the disambiguation technique, compared to the state of the art Random-Walk model.

Malin (2005) investigates unsupervised methods which simultaneously learn the number of entities represented by a particular name and which observations correspond to the same entity. The disambiguation methods leverage the fact that an entity's name can be listed in multiple sources, each with a number of related entity's names, which permits the construction of name-based relational networks. The author proposes two different methods which differ based on the type of network similarity exploited for disambiguation. The first method relies upon exact name similarity and employs hierarchical clustering of sources, where each source is considered a local network and represented as a boolean vector $s_i = [e_{i1}, \ldots, e_{in}]$, where $e_{ij} = 1$ if e_j is in source s_i and 0 otherwise. In contrast, the second method employs a less strict similarity requirement by using random walks between ambiguous observations on a global social network constructed from all sources, or a community similarity. The graph has been built with putting a node for every distinct name in S and an edge between two nodes if the names collocate in a source at least one time. Experiments have been conducted on a subset of the Internet Movie Database. Results demonstrate that community equivalence (the second method) provides an advantage over exact equivalence (first method) for

measuring similarity and, subsequently, disambiguation.

Graph-based models have also been applied to person name disambiguation, usually benefiting from the social networks in people related tasks. Minkov et al. (2006) consider extended similarity metrics for documents and other objects embedded in graphs, implemented via a lazy graph walk. They provide an instantiation of this framework for email data, where content, social networks and a timeline are integrated in a structural graph. The suggested framework is evaluated for two email-related problems: disambiguating names in email documents, and threading. Resolving the referent of a person name is also an important complement to the ability to perform named entity extraction for tasks like social network analysis or studies of social interaction in email. The authors model this problem as a search task: based on a name-mention in an email message m, they formulate query distribution V_q , and then retrieve a ranked list of person nodes. Experiments carried out on the Cspace corpus [Minkov et al. (2005)], manually annotated with personal names, show that reranking schemes based on the graph-walk similarity measures often outperform baseline methods, with a maximum accuracy of 83.8%.

2.3 Open Access Resources

In this section will be provided a description of resources exploited within this work. It will be described the organization of WordNet, of Wikipedia and will be presented a solution for incapsulating lexical and semantic knowledge.

2.3.1 WordNet

WordNet [Miller (1995)] is a semantic lexicon for English language. It groups English words into sets of synonyms called synsets. It provides short, general definitions and it records the various semantic relations between these synonym sets (but also includes lexical relations between words). Every synset contains a group of synonymous words or collocations (a collocation is a sequence of words that go together to form a specific meaning, such as “car pool”); different senses of a word are in

different synsets, given that each synset represents one underlying lexical concept. The meaning of the synsets is further clarified with short defining glosses. Different relations link the synonym sets. These relations can be used to engineer a change of representation in text data by transforming vectors of words into vectors of word meanings. The synonymy relation can be used to map words with similar meanings together, and hyperonymy (corresponding to the "is-a" relation) can be used to generalize noun and verb meanings to a higher level of abstraction. In the following paragraphs, we describe the main features of WordNet.

The origin of WordNet In 1985 a group of psychologists and linguists at Princeton University undertook to develop a lexical database along lines suggested by investigations in psycho-lexicology, the research concerned with the lexical component of language. The initial idea was to provide an aid in searching dictionaries conceptually, rather than merely alphabetically. It was to be used in close conjunction with an on-line dictionary of the conventional type. As the work proceeded, however, it demanded a more ambitious formulation of its own principles and goals. WordNet is the result.

The most ambitious feature of WordNet, however, is its attempt to organize lexical information in terms of word meanings, rather than word forms. In that respect, WordNet resembles a thesaurus more than a dictionary, even if WordNet is not merely an on-line thesaurus, however. In order to appreciate what more has been attempted in WordNet, it is necessary to understand its basic design.

The Lexical Matrix Lexical semantics begins with a recognition that a word is a conventional association between a lexicalized concept and an utterance that plays a syntactic role. This definition of *word* raises at least three classes of problems for research. First, what kinds of utterances enter into these lexical associations? Second, what is the nature and organization of the lexicalized concepts that words can express? Third, what syntactic roles do different words play? Although it is impossible to ignore any of these questions while considering only one, the emphasis here will be on the second class of problems, those dealing with the semantic structure of the English lexicon.

Word Meanings	**Word Forms**			
	F_1	F_2	...	F_n
M_1	$E_{1,1}$	$E_{1,2}$		
M_2		$E_{2,2}$		
...				
M_m				$E_{m,n}$

Table 2.2: The Lexical Matrix: F_1 and F_2 are synonyms; F_2 is polysemous

Since the word "word" is commonly used to refer both to the utterance and to its associated concept, discussions of this lexical association are vulnerable to terminological confusion. In order to reduce ambiguity, therefore, "word form" will be used here to refer to the physical utterance or inscription and "word meaning" to refer to the lexicalized concept that a form can be used to express. Then the starting point for lexical semantics can be said to be the mapping between forms and meanings. A conservative initial assumption is that different syntactic categories of words may have different kinds of mappings. Word forms are imagined to be listed as headings for the columns; word meanings as headings for the rows. An entry in a cell of the matrix (Table 2.2) implies that the form in that column can be used (in an appropriate context) to express the meaning in that row. Thus, entry $E_{1,1}$ implies that word form F_1 can be used to express word meaning M_1. If there are two entries in the same column, the word form is polysemous; if there are two entries in the same row, the two word forms are synonyms (relative to a context).

Mappings between forms and meanings are *many to many*. Some forms have several different meanings, and some meanings can be expressed by several different forms. Two difficult problems of lexicography, *polysemy* and *synonymy*, can be viewed as complementary aspects of this mapping. That is to say, polysemy and synonymy are problems that arise in the course of gaining access to information in the mental lexicon: a listener or reader who recognizes a form must cope with its polysemy; a speaker or writer who hopes to express a meaning must decide between synonyms.

How are word meanings represented in WordNet? In order to simulate a lexical matrix it is necessary to have some way to represent both forms and meanings in a computer. Inscriptions can provide a reasonably sat-

isfactory solution for the forms, but how meanings should be represented poses a critical question for any theory of lexical semantics. Lacking an adequate psychological theory, methods developed by lexicographers can provide an interim solution: definitions can play the same role in a simulation that meanings play in the mind of a language user. If the person who reads the definition has already acquired the concept and needs merely to identify it, then a synonym (or near synonym) is often sufficient. In other words, the word meaning M_1 in Table 2.2 can be represented by simply listing the word forms that can be used to express it: $\{F_1, F_2, ...\}$ (here and later, the curly brackets, '{' and '}', surround the sets of synonyms that serve as identifying definitions of lexicalized concepts). For example, someone who knows that *board* can signify either a piece of lumber or a group of people assembled for some purpose will be able to pick out the intended sense with no more help than plank or committee. The synonym sets, {board, plank} and {board, committee} can serve as unambiguous designators of these two meanings of board.

The synonym sets, called *synsets*, do not explain what the concepts are; they merely signify that the concepts exist. People who know English are assumed to have already acquired the concepts, and are expected to recognize them from the words listed in the synset. A lexical matrix, therefore, can be represented for theoretical purposes by a mapping between written words and synsets. Sometimes, however, an appropriate synonym is not available, in which case the polysemy can be resolved by a short gloss, e.g., {board, (a person's meals, provided regularly for money)} can serve to differentiate this sense of board from the others; it can be regarded as a synset with a single member. The gloss is not intended for use in constructing a new lexical concept by someone not already familiar with it, and it differs from a synonym in that it is not used to gain access to information stored in the mental lexicon. It fulfills its purpose if it enables the user of WordNet, who is assumed to know English, to differentiate this sense from others with which it could be confused.

Synonymy is, of course, a lexical relation between word forms, but because it is assigned this central role in WordNet, a notational distinction is made between words related by synonymy, which are enclosed in curly brackets, '{' and '}', and other lexical relations, which will be enclosed in

square brackets, '[' and ']'. Semantic relations are indicated by pointers. WordNet is organized by semantic relations. Since a semantic relation is a relation between meanings, and since meanings can be represented by synsets, it is natural to think of semantic relations as pointers between synsets. It is characteristic of semantic relations that they are reciprocated: if there is a semantic relation R between meaning $\{x, x', ...\}$ and meaning $\{y, y', ...\}$, then there is also a relation R' between $\{y, y', ...\}$ and $\{x, x', ...\}$.

Synonymy From what has already been said, it should be obvious that the most important relation for WordNet is similarity of meaning, since the ability to judge that relation between word forms is a prerequisite for the representation of meanings in a lexical matrix. Two expressions could be considered as synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made. By that definition, true synonyms are rare, if they exist at all. A weakened version of this definition would make synonymy relative to a context: two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value. Note that the definition of synonymy in terms of substitutability makes it necessary to partition WordNet into nouns, verbs, adjectives, and adverbs. That is to say, if concepts are represented by synsets, and if synonyms must be interchangeable, then words in different syntactic categories cannot be synonyms (cannot form synsets) because they are not interchangeable. Nouns express nominal concepts, verbs express verbal concepts, and modifiers provide ways to qualify those concepts. In other words, the use of synsets to represent word meanings is consistent with psycholinguistic evidence that nouns, verbs, and modifiers are organized independently in semantic memory.

Hyponymy/Hyperonymy Hyponymy/hyperonymy is a semantic relation between word meanings: e.g., {maple} is a hyponym of {tree}, and {tree} is a hyponym of {plant}. Much attention has been devoted to *hyponymy/hyperonymy* (variously called subordination/superordination, subset/superset, or the IS-A relation). A concept represented by the synset $\{x, x', ...\}$ is said to be a hyponym of the concept represented by

the synset $\{y, y', ...\}$ if native speakers of English accept sentences constructed from such frames as "An x is a (kind of) y". The relation can be represented by including in $\{x, x', ...\}$ a pointer to its superordinate, and including in $\{y, y', ...\}$ pointers to its hyponyms.

Hyponymy is transitive and asymmetrical, and, since there is normally a single superordinate, it generates a hierarchical semantic structure, in which a hyponym is said to be below its superordinate. Such hierarchical representations are widely used in the construction of information retrieval systems: a hyponym inherits all the features of the more generic concept and adds at least one feature that distinguishes it from its superordinate and from any other hyponyms of that superordinate. For example, *maple* inherits the features of its superordinate, *tree*, but is distinguished from other trees by the hardness of its wood, the shape of its leaves, the use of its sap for syrup, etc. This convention provides the central organizing principle for the nouns in WordNet. Figure 2.3 shows the output of WordNet when a user requests all the hypernyms for the noun senses of the word *bat*. WordNet contains 5 noun senses for *bat*, and each corresponding synset is displayed, followed by all the synsets that appear above it in the hypernym hierarchy. The figure shows only the hierarchy of sense 1.

```
bat, chiropteran -- (nocturnal mouselike mammal with
forelimbs modified to form membranous wings and anatomical
adaptations for echolocation by which they navigate)
   => placental, placental mammal, eutherian, eutherian mammal
   -- (mammals having a placenta; all mammals except monotremes
   and marsupials)
       => mammal -- (any warm-blooded vertebrate having the skin
       more or less covered with hair; young are born alive except
       for the small subclass of monotremes and nourished with milk)
           => vertebrate, craniate -- (animals having a bony or
           cartilaginous skeleton with a segmented spinal column and
           a large brain enclosed in a skull or cranium)
               => chordate -- (any animal of the phylum Chordata
               having a notochord or spinal column)
                  => animal, animate being, beast, brute, creature,
                  fauna -- (a living organism characterized by
                  voluntary movement)
                     => organism, being -- (a living thing
                     that has (or can develop) the ability to act
                     or function independently)
                         => living thing, animate thing -- (a
                         living (or once living) entity)
                             => object, physical object -- (a
                             tangible and visible entity; an
                             entity that can cast a shadow; "it
                             was full of rackets, balls and
                             other objects")
                                 => entity -- (that which is
                                 perceived or known or inferred to
                                 have its own distinct existence
                                 (living or nonliving))
```

Figure 2.3: The hierarchy of sense 1 of the word "bat" obtained from WordNet (version 1.7.1)

2.3.2 Wikipedia

An important role in NLP is played by Wikipedia which is a free, multilingual, open content encyclopedia project. Its name is a portmanteau

of the words wiki (a technology for creating collaborative websites) and encyclopedia: articles have been written collaboratively by volunteers around the world. Wikipedia has been launched in 2001 by Jimmy Wales and Larry Sanger and now is the most popular general reference work currently available on the Internet. Wikipedia has steadily gained status as a general reference website and its content has also been used in academic studies, books and conferences[11]. Many studies that try to exploit Wikipedia as a knowledge source have recently emerged [Ponzetto and Strube (2006); Strube and Ponzetto (2006); Zesch et al. (2007)]. In particular, for the problem of entities Bunescu and Pasca exploited internal links in Wikipedia as training examples [Bunescu and Pasca (2006)], while Cucerzan used Wikipedia to extract entities to validate his proposal [Cucerzan (2007)]. Although it cannot be used as gazetteers directly since it is not intended as a machine readable resource, extracting knowledge such as gazetteers from Wikipedia it is easier than from raw texts or from usual Web texts because of its structure [Toral and Munoz (2006)].

Wikipedia open nature gives rise to criticisms concerning reliability and accuracy, such as the addition of spurious or unverified information; uneven quality, systemic bias and inconsistencies. To ensure the quality, the project relies on its community members to remove vandalism or identify problems such as violation of neutrality and factual errors in its articles. Articles in Wikipedia are subject to several policies and guidelines; they need to be on "notable" topics, contain "no original research" and only "verifiable" facts and must be written from a "neutral point of view". Contents that fail to meet those guidelines and policies are modified or get deleted. Giles in his famous article “Internet encyclopaedias go head to head” demonstrates the accuracy of Wikipedia against the solid Britannica Encyclopedia [Giles (2005)].

Almost every article in Wikipedia may be edited anonymously through a user account, while only registered users may create a new article. Since Wikipedia aims to be an encyclopedia, each article is structured as the explaination of a proper concept, the real world object the article is describing and for this reason most articles represent a Named Entity. Any individual topic within Wikipedia is called *Page*, which consist of the web

[11] http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_in_academic_studies

page without the top, bottom and sidebars. Pages include articles, stubs, redirects, disambiguation pages, user pages, talk pages, documentation and special pages.

The many-to-many correspondence between names and entities is captured in Wikipedia through redirect and disambiguation pages.

A *redirect page* exists for each alternative name that can be used to refer to an entity in Wikipedia. A redirect is a page which has no content itself, but sends the reader to another article or page, usually from an alternative title. The name is transformed (using underscores for spaces) into a title whose article contains a redirect link to the actual article for that entity. It is used for synonyms. For example, *impressionist* might redirect to *impressionism*.

Disambiguation pages are created for ambiguous names, i.e. names that denote two or more entities in Wikipedia. A disambiguation page contains various meanings of a word, and refers to the pages where the various meanings are defined. In cases when there is a prevailing meaning of the term, disambiguation pages are named "subject (disambiguation)".

Every article in Wikipedia is required to have at least one *category*. Categories allow articles to be placed into one or more topics. These topics can be further categorized by associating them with one or more parent categories. A category is a collection of pages automatically formed by the Wikipedia servers by analyzing category tags in articles. Category tags are in the form Category:Computers. The part after the ":" is the name of the Category. Adding a category tag causes a link to the category and any super-categories to go to the bottom of the page. As stated, it also results in the page being added to the category listing.

Another important feature of Wikipedia is linking through hyperlinks. Articles in Wikipedia often contain mentions of entities that already have a corresponding article. When contributing authors mention an existing Wikipedia entity inside an article, they are required to link at least its first mention to the corresponding article, by using *links* (the title of referred entity within double square brackets [[title]]) or piped links (the title of referred entity followed by a different display name within double square brackets [[title | display name]]). Internal links bind Wikipedia into an interconnected whole. Links can also be external to Wikipedia: the aim is to provide instant pathways to locations within and outside Wikipedia

that are likely to increase understanding of the topic at hand.

2.3.3 Lexical Browser

Lexical Browser is the web interface to Lexical Collector, a service that, given a lexeme (an abstract unit of morphological analysis in linguistics, which roughly corresponds to a set of words that are different forms of the same word), returns all syntactic and semantic information collected from a list of lexical and semantic resources [Gentile et al. (2008)]. The proposed strategy consists in merging data with origin from stable resources, such as WordNet, with data collected dynamically from evolving sources, such as the Web or Wikipedia. The strategy is implemented in a wrapper to a set of popular linguistic resources that provides a single point of access to them, in a transparent way to the user, to accomplish the computational linguistic problem of getting a rich set of linguistic and semantic annotations in a compact way. The Collector system encompasses a set of resources, such as WordNet, WordNet Domains, WordNet Affect, WordNet Clustering, WordNet Mapping and Topinc Signitures as static resources, whether it queries the web and Wikipedia for dynamic information. While WordNet and Wikipedia have been exhaustively described in section 2.3.1 and 2.3.2, in the following paragraphs will be given a brief overview of other encapsulated resources. ’

Encapsulated Resources

Web Search The World Wide Web is a mine of language data of unprecedented richness and ease of access [Kilgarriff and Grefenstette (2003)] and it has been used and proved to be useful in many linguistic tasks, spreading from collecting frequency data [Turney (2001)], to the building of corpus both with specific features [Ghani et al. (2001)] or even with a general large scale scope [Terra and Clarke (2003)]. Question Answering systems can also take advantage from the Web as a source of answers [Radev et al. (2001); Kwok et al. (2001)] or validation [Magnini et al. (2002)]. The list can also continue with Word Sense Disambiguation, where the Web was proved to be more effective when integrated with other resources rather than stand-alone [Rosso et al. (2005)]. For this reasons

we decided to integrate in our system a service that simply find out the first n links provided by a web search engine for using the lexeme as query. As web search engine we used Yahoo! Search APIs [12].

WordNet related Resources The system includes a set of WordNet related resources, such as WordNet Domains, WordNet Affect, WordNet Clustering, Topic Signitures and WordNet Mapping.

WordNet Domains [Magnini and Cavaglià (2000)] provides semantic domains, as a natural way to establish semantic relations among word senses, which can be profitably used for Computational Linguistics. Semantic domains are areas of human discussion, such as POLITICS, ECONOMY, SPORT, which exhibit their own terminology and lexical coherence. WordNet Domains was created by augmenting the Princeton English WordNet with domain labels. Information brought by domains is complementary to what is already in WordNet. A domain may include synsets of different syntactic categories and from different WordNet sub-hierarchies. Domains may group senses of the same word into homogeneous clusters, with the side effect of reducing word polysemy in WordNet. We use WordNet Domains 2.0 developed by TCC Division of ITC-irst (Trento-ITALY).

WordNet Affect [Strapparava and Valitutti (2004)] is an affective lexicon and was developed as an extension of WordNet in order to collect affective concepts and correlate them with words. We use the version 1.0 of WordNet Affect which refers to synsets in WordNet 1.6. Each synset is labeled with a particular 'affective-state' such as emotion, cognitive state, hedonic signal, emotional response and so on. The resource contains *core synsets* that are manually labeled from expert and other synset obtained using WordNet relation like *similar-to* and *antonym*.

WordNet clustering is a method for reducing the granularity of the WordNet sense inventory based on the mapping to a manually crafted dictionary encoding sense hierarchies, namely the Oxford Dictionary of English. We use WordNet clustering based on WordNet 2.1. A detailed description of WordNet Clustering can be found in [Navigli (2006)].

Another included resource is Topic Signature [Agirre and Lopez de Lacalle Lekuona (2004)], that is a list of words that co-occur with the

[12] http://developer.yahoo.com/faq/

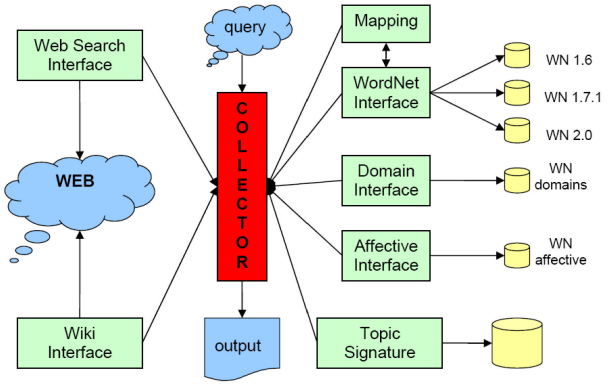

Figure 2.4: Lexical Browser System architecture

words in a particular synset in WordNet automatically built using the Web. The topic signatures are built for the word senses of english nouns (monosemous and polysemous) in WordNet version 1.6.

Our system also provides a mapping between synsets offsets in various WordNet versions. The current supported mappings are:

1. from WordNet 1.5 to WordNet 1.6
2. from WordNet 1.6 to WordNet 1.7.1
3. from WordNet 1.7.1 to WordNet 2.0
4. from WordNet 2.0 to WordNet 2.1

System Architecture The system architecture is sketched in figure 2.4. The main component is the Collector that processes the user query and collects all informations over different resources. The system has two kinds of interfaces, one for what we called *static resource* (in figure 2.4, at the right side of the main component, Collector) and one for the *dynamic ones* (in figure 2.4, at the left side). In particular, at the current state of the work, the dynamic side consists of two interfaces that allow to query the Web and Wikipedia (respectively Web Search Interface and Wiki Interface). On the other hand, the static side consists of interfaces that supply the access to local resources, in detail:

- Mapping and WordNet interfaces provide the access respectively to WordNet mapping and different versions of WordNet;
- Domain Interface supplies the WordNet domain informations;

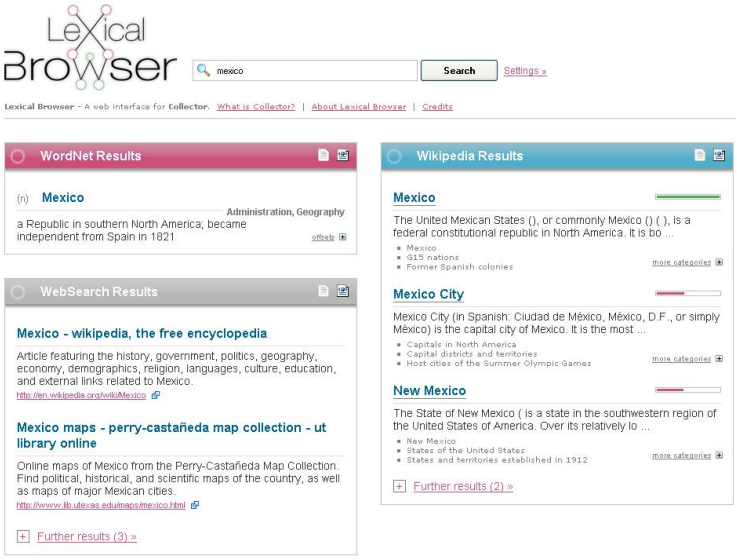

FIGURE 2.5: Search results of the Lexical Browser

- Affective Interface provides the access to WordNet Affective;
- Topic Signature Interface retrieves a topic signatures for each synset.

Finally the output is a XML document that contains all the collected information.

The flexibility of the Collector Architecture permits to plug new resources in a simple way: the only effort consists of providing the actual resource and to implement an Interface that performes the effective access to that. To plug a new resource, for instance a domain ontology, we need to develop xyzOntology Interface for defining relevant information of the ontology that we need to extract and for designing the structure of the XML node we want to append to the final document.

Lexical Browser Lexical Browser is a web application, developed using the Java technology, that provides a web access to Lexical Collector. Given a lexeme, Lexical Browser invokes the Collector to obtain an XML document containing all information about the lexeme; furthermore the web application permits to obtain the XML document or part of it, through the graphic interface or through web services.

It also gives the possibility to store information about users and their searches to collect statistics.

A demo version of the Lexical Browser (Figure 2.5) is available at the URL *http://193.204.187.223:8080/LexicalBrowser/*.

System Output Elaboration results are stored in an XML document. The root node (tagged as Entry) has the attribute Id valued with the lexeme. Each child node contains information obtained quering employed resources using the lexeme.

The information derived from WordNet concern offset of the synset in different versions of WordNet, single words, domain and gloss.

```
<wnentry posTag="2">
 <offsets>
   <offset id="n07739854" version="1"/>
   <offset id="n09157420" version="2"/>
   <offset id="n10436898" version="3"/>
   <offset id="n10998387" version="4"/>
 </offsets>
 <words>
   <word>Mary</word>
   <word>Virgin_Mary</word>
   <word>the_Virgin</word>
   <word>Madonna</word>
 </words>
 <domains>
   <domain>person</domain>
   <domain>religion</domain>
 </domains>
 <gloss>
   The mother of Jesus; Christians refer to her as the
   Virgin Mary; she is especially honored by Roman Catholics
 </gloss>
</wnentry>
```

For each Wikipedia entry, Lexical Collector crawls data about URL, title, a short definition for the content of the page.

```
<wikientry url="..."
title="Blessed_Virgin_Mary">
  <definition text="This article is about the Catholic,
  Orthodox and Anglican understanding of Mary; for other
  views, see Mary (mother of Jesus) and Islamic view of Virgin Mary.../>
  ...
</wikientry>
```

Additionally there are data about categories related to the page, all captions of images, all related pages within Wikipedia and external links

cited within the page.

```
<categories>
  <category label="Titles of Mary"/>
</categories>

<imageCaptions>
  <imageCaption text="Immaculate Heart"/>
  <imageCaption text="Our Lady of Guadalupe.
   Highly venerated image in Mexico."/>
  <imageCaption text="Fedorovskaya"/>
  ...
</imageCaptions>

<relatedPages>
  <relatedPage reference=
   "Mary (mother of Jesus)"/>
  <relatedPage reference=
   "Islamic view of Virgin Mary"/>
  <relatedPage reference=
   "Immaculate Heart of Mary"/>
  ...
</relatedPages>

<webLinks/>
```

Chapter 3

Named Entity Disambiguation Algorithms

This chapter presents a detailed description of NED algorithms proposed in this dissertation. Before introducing the description of the algorithms, some formal definitions are needed. The Named Entity Disambiguation problem will be treated as a special case of Word Sense Disambiguation.

Definition 10 *The goal of a WSD algorithm consists in assigning a word w_i occurring in a document d with its appropriate meaning or sense s. The sense s is selected from a predefined set of possibilities, usually known as sense inventory. Generally, a WSD algorithm takes as input a document $d = \{w_1, w_2, \ldots, w_h\}$ and returns a list of senses $X = \{s_1, s_2, \ldots, s_k\}$ in which each element s_j is obtained by disambiguating the target word w_i. Notice that $k \leq h$ because some words, such as proper names, might not be found in sense inventory or a sequence of words can be identified as a multi-words expression, for example artificial intelligence.*

Rephrasing definitions 2-9 (Chapter 2) and drawing on definition 10, the following is established.

Definition 11 *The goal of a NED algorithm consists in associating a reference of an entity e_i occurring in a document d with its appropriate identity (the referent or meaning or sense) s, that is the correct unique real-world object the occurrence refers to. The referent s is selected from a predefined set of possibilities, that will be referred as entity inventory.*

We would denominate the occurrence of an entity in the the text with the terms *reference* or *name* and the addressed real world object with the term *referent*, following name convention given in chapter 1. It would be assumed to have an *entity inventory*, containing all possible referents, and that each reference has associated the set of candidate entities, that is all real-world objects that a single reference could refer.

A NED algorithm can adopt several strategies to solve the name ambiguity.
In this chapter will be proposed two different NED algorithms:

- $WiBNED$: a knowledge-based NED algorithm based on the idea of exploiting a freely available Knowledge Source, Wikipedia, to solve the Named Entity Disambiguation problem.
- $SRaNED$: a graph-based algorithm which exploits Semantic Relatedness measures over Wikipedia to provide a solution for the NED task.

In Section 3.3 it will be described META, *(MultilanguagE Text Analyzer)*, a tool for text analysis which implements some Natural Language Processing functionalities, including NED.

3.1 WibNED: Wikipedia based Named Entity Disambiguation

As stated in definition 10 the goal of a WSD algorithm consists in assigning a word w_i occurring in a document d with its appropriate meaning or sense s. Usually, WSD algorithms make use of the *context* C in which w_i is found. The context C for w_i is defined as a set of words that precede and follow w_i.
The Lesk algorithm is a classical algorithm for WSD for all words in unrestricted text. It was introduced by Lesk (1986). The basic assumption is that words in a given neighbourhood will probably share a common

topic. Apart from knowledge about the context (the immediate surrounding words), the algorithm requires a machine readable dictionary, with an entry for each possible sense for a word. The original algorithm takes into account words pairwise and computes overlap among sense definitions: the sense pair with the highest overlap score is chosen.

The proposed algorithm, named WibNED [Gentile et al. (2009)], is an adaptation of Lesk dictionary-based WSD algorithm [Banerjee and Pedersen (2002)]. In the WibNED algorithm the words to disambiguate are only those representing an Entity.
WibNED takes as input a document $d = \{w_1, \ldots, w_j, e_{j+1}, w_{j+2}, \ldots, w_h, e_{h+1}, w_{h+2}, \ldots\}$ and returns a list of Wikipedia URIs $X = \{s_1, s_2, \ldots, s_k\}$ in which each element s_i is obtained by disambiguating the *target entity* e_i on the ground of the information obtained from Wikipedia for each candidate URI (Wikipedia page content of the URI) and words in the *context* C of e_i. The *context* C of the target entity e_i is defined to be a window of n words to the left and another n words to the right, for a total of $2n$ surrounding words. In the current version of the algorithm if other entities occur in the context of the target entity, they are considered as words and not as entities.

Algorithm 1 describes the complete procedure for the disambiguation of entities. The input for the algorithm is an ordered list W

$$W = (w_1, \ldots, w_j, e_{j+1}, w_{j+2}, \ldots, w_h, e_{h+1}, w_{h+2}, \ldots)$$

of words in a document, processed with a NLP tool: w_i are common words while e_i are tagged as Named Entities.

The NLP tool used to process the document collection is META [Basile et al. (2008d)], further described in Section 3.3, that has been used to perform the following operations:

- tokenization;
- part of speech (POS) tagging;
- stop words elimination;
- WSD by using WordNet;

- Entity Recognition performed with Yamcha [Kudo and Matsumoto (2003)], a NER annotator which uses a Support Vector Machine techniques.

Each document of the collection is then transformed in an ordered list of common words w_i and Named Entities e_i:

$$W = (w_1, \ldots, w_j, e_{j+1}, w_{j+2}, \ldots, w_h, e_{h+1}, w_{h+2}, \ldots)$$

WibNED

The list W is the input for the main procedure, named WibNED, that finds the proper Wikipedia URI for each polysemous entity e_i in W. WibNED uses several subprocedures.

QueryWiki

The subprocedure QueryWiki finds all possible Wikipedia pages answering the query e_i. Each element of the returning set contains the page URI and a short textual description of the page.

PickPage

The subprocedure pickPage computes the overlap between the context of the target entity and the description obtained from the Wikipedia page for each candidate sense. It returns the candidate entity which maximizes the overlap.

The evaluation of WiBNED algorithm is reported in the Chapter 4.

Algorithm 1 *WibNED*, the algorithm for the disambiguation of entities

procedure $WibNED(W)$
▷ finds the proper Wikipedia URI for each polysemous entity e_i in the ordered list $W = (w_1, \ldots, w_j, e_{j+1}, w_{j+2}, \ldots, w_h, e_{h+1}, w_{h+2}, \ldots)$
W is the list of words in a document, processed with a NLP tool.
w_i are common words while e_i are tagged as Named Entities.
  $S \leftarrow W$
  for all $e_i \in S$ **do**
    $Context_{e_i} \leftarrow \{w_{i-n}, ..., w_{i+n}\}$
    $Candidate_{e_i} \leftarrow QueryWiki(e_i)$
    $s_{e_i} \leftarrow pickPage(Context_{e_i}, Candidate_{e_i})$
    $S.replace(e_i, s_{e_i})$
  end for
  return S
end procedure

procedure $QueryWiki(e_i)$
▷ finds all possible Wikipedia pages answering the query e_i. Each element of the returning set contains the page URI and a short textual description of the page.
  $Candidate \leftarrow \{\}$
  $WikiResults \leftarrow allpagesfromWikipediaansweringthequery e_i$
  for all $wikis_i \in WikiResults$ **do**
    $c_i.uri \leftarrow wikis_i$
    $c_i.description \leftarrow describe(wikis_i)$
▷ $describe(wikis_i)$ builds a short textual description of the page.
    $Candidate.add(c_i)$
  end for
  return $Candidate$
end procedure

procedure $pickPage(Context, Candidate)$
  $maxOverlap \leftarrow 0$
  $bestPage \leftarrow$ **null**
  for all $c_i \in Candidate$ **do**
    $currentOverlap \leftarrow computeOverlap(c_i.description, Context)$
    if $currentOverlap > maxOverlap$ **then**
▷ $computeOverlap(c_i.description, Context)$ calculate the number of overlapping words between $Context$ and $c_i.description$
      $bestPage \leftarrow c_i.uri$
      $maxOverlap \leftarrow currentOverlap$
    end if
  end for
  return $bestPage$
end procedure

3.2 SRaNED: Semantic Relatedness approach for Named Entity Disambiguation

The second method proposed is SRaNED, whose name stands for Semantic Relatedness approach for Named Entity Disambiguation. Similarly to WibNED, it is a Knowledge-based method, exploiting Wikipedia as freely available Knowledge Base. The contributions given from SRaNED are twofold. Firstly, it uses a random-walk model based Semantic Relatedness approach to NED. Graph-based models, as illustrated in Chapter 2, have previously been applied to WSD [Agirre et al. (2006); Mihalcea (2005); Navigli and Lapata (2007); Sinha and Mihalcea (2007)] but not experimented for the problem of NED: to the best of our knowledge, previous approaches to NED were based on Vector Space model, treating *concepts* and context texts as a bag of words [Bunescu and Pasca (2006); Cucerzan (2007)], while graph-based models have been exploited for a specific type of NED, which is Person Name Disambiguation [Minkov et al. (2006)], or for specific domains, such as bibliographic citations [Kalashnikov and Mehrotra (2005)]. The solution proposed with SRaNED exploits Semantic Relatedness Scores (calculated with a random walk on a graph) as input for disambiguation step. Secondly, the proposed solution introduce a different way for representing the context for the target entity which, rather than consisting of surrounding words, is composed of only other named entities present in the text. The approach has the advantage of using relatedness scores independently for the NED task, that means using semantic relations as input for NED: this is useful in terms of clearly separation of two different functions, which can be refined and ameliorated separately one from each other. Compared to the best result by Cucerzan [Cucerzan (2007)], which is an accuracy of 91.4% and of 88.3%, this method achieves a competitive accuracy of 91.46 % and 89.83% respectively, as it will be shown in Chapter 4, and it adds the benefit of having two clearly separate steps (relatedness sores, disambiguation), thus providing a glimpse of improving results in both directions.

Given a set of *surfaces* and their corresponding concept relatedness matrix R, the SRaNED algorithm returns for each *surface* one *sense*

(*concept*), that is collectively determined by other *surfaces* and their corresponding *concepts*. To achieve this goal the proposed method performs four main sequential steps: 1) each text is reduced to the list of *surfaces* of appearing entities; 2) for each *surface*, Wikipedia is used to retrieve all its possible *meanings* (also denoted as *concepts*) and build a feature space for each of them; 3) all *concepts*, their features and relations are transformed into a graph representation: a random graph walk model is then applied to combine the effects of features and derive a relatedness score; 4) for each *surface* a single *meaning* is chosen, taking into account *Semantic Relation* within the entity graph.

Concept Retrieval

In more details, as a starting point for the proposed methodology it is assumed that each text has been reduced to the list of its contained *named entity surfaces*, as it is simply obtainable with a standard NER system, as e.g Yamcha [Kudo and Matsumoto (2003)]. Then for each *surface*, Wikipedia is used to retrieve all its possible *meanings* and build a feature space for each of them. More precisely each *surface* is used to query Wikipedia and retrieve relevant pages. If a *surface* matches an entry in Wikipedia, a page will be returned. If the *surface* has only one sense defined in Wikipedia then we have a single result: the page describing the concept that matches the surface form. This page is referred as the *sense page* for the concept. Alternatively a *disambiguation page* may be returned if the *surface* has several *senses* defined in Wikipedia. Such a page lists different senses as links to other pages and with a short description for each one. For the purpose of this method, for every *surface* the disambiguation page has been deliberately chosen to follow every link on the page and keep all *sense pages* for that surface. This is done by appending the keyword "(disambiguation)" to a *surface* as a query. Thus, for every *surface*, we obtain a number of *concepts* (represented as *sense pages*) as input to SRaNED algorithm. Once relevant *concepts* and their *sense pages* for the input *concept surface forms* have been identified, the *sense page* retrieved from Wikipedia for each *concept* are used to build its feature space. The following features have been figured out as potentially useful:

1. Words composing the titles of a page *(title_ words)*: words in the title of a sense page; plus words from all its redirecting links in Wikipedia (different *surfaces* for the same concept).
2. Top n most frequently used words in the page *(frequent_ words_ n)*: prior work makes use of words extracted from the entire page [Strube and Ponzetto (2006)], or only those from the first paragraph [Zesch et al. (2008)]. For the purpose of this work, the most frequent words have been used, based on the intuition that word frequency indicates the importance of the word for representing a topic.
3. Words from categories *(cat_ words)* assigned to the page: each page in Wikipedia is assigned several category labels. These labels are organized as a taxonomy. The category labels assigned to a page are retrieved by performing a depth limited search of 2, and splitting these labels to words.
4. Words from outgoing links on the page *(link_ words)*: the intuition is that links on the page are more likely to be relevant to the topic, as suggested by Turdakov and Velikhov [Turdakov and Velikhov (2008)].

Thus, for each *concept* all the above features are extracted from its sense page, and then the text features are transformed into a graph conforming to the random walk model, which is used to compute Semantic Relatedness between meanings belonging to different surfaces.

Random Graph Walk Model

A random walk is a formalization of the intuitive idea of taking successive steps in a graph, each in a random direction [Lovász (1996)]. Intuitively, the harder it is to arrive at a given node starting from another, the less related the two nodes are. The advantage of a random-walk model lies at its strength of seamlessly combining different features to arrive at one single measure of relatedness between two entities [Iria et al. (2007)]. The built undirected weighted typed graph encompasses all concepts identified in the page retrieval step and their extracted features. The graph is a 5-tuple G = (V, E, t, l, w), where V is the set of nodes representing the concepts and their features; E: $V \times V$ is the set of edges that connect concepts and their features, representing an undirected path from

concepts to their features, and vice versa; $t\colon V \to T$ is the node type function ($T = \{t_1, \ldots, t_{|T|}\}$ is a set of types, e.g. concepts, *title_ words*, *cat_ words*, ...), $l\colon E \to L$ is the edge label function ($L = \{l_1, \ldots, l_{|L|}\}$ is a set of labels that define relations between concepts and their features), and$w\colon L \to R$ is the label weight function that assigns a weight to an edge. Figure 3.1 shows a piece of the graph with types and labels described before: circles indicate nodes (V) representing concepts and features; bold texts indicate types (T) of nodes; solid lines connecting nodes indicate edges (E), representing relations between concepts and features; italic texts indicate types (L) of edges. Concepts sharing same features will be connected via the edges that connect features and concepts.

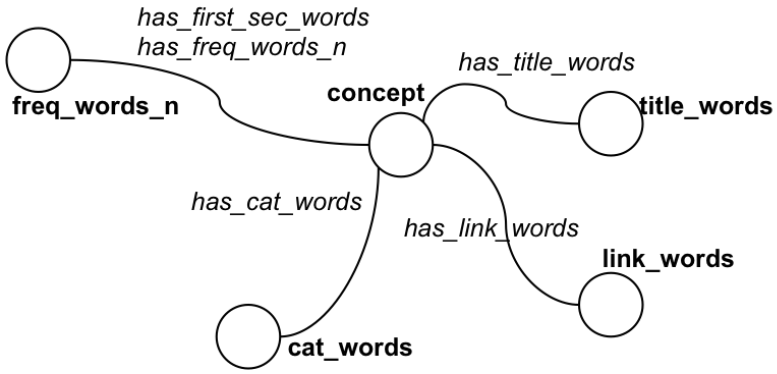

Figure 3.1: The Graph representation model of concepts, features, and their relations

Weights are defined for each edge type, which, informally, determine the relevance of each feature to establish the relatedness between any two concepts. Let $L_{t_d} = l(x, y)\colon (x, y) \in E \cap T(x) = t_d$ be the set of possible labels for edges leaving nodes of type t_d. It is required that the weights form a probability distribution over L_{t_d} , i.e.

$$\sum_{\in L_{t_d}} w(l) = 1$$

An adjacency matrix of locally appropriate similarity between nodes is built as

$$W_{ij} = \begin{cases} \sum_{l_k \in L} \frac{w(l_k)}{|(i,\cdot) \in E\colon l(i,\cdot) = l_k|} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where W_{ij} is the i^{th}-row and j^{th}-column entry of W, indexed by V. The above equation distributes uniformly the weight of edges of the same type leaving a given node. To tune the weights of different features a simulated annealing optimization method has been used, as described in [Nie et al. (2005)]. To simulate the random walk, it has been applied a matrix transformation using the formula $P^{(t)}(j \mid i) = [(D^{-1}W)^t]_{ij}$, as described by Iria et al. (2007), where D is the diagonal degree matrix given by $D_{ii} = \sum_k W_{ik}$, and t is a parameter representing the number of steps of the random walk. For the aim of this work, it has been set $t = 2$ to prevent smoothing out the graph . The resulting matrix of this transition $P^{(t)}(j \mid i)$ is a sparse, non-symmetric matrix filled with probabilities of reaching node i from j after t steps. To transform probability to relatedness, it has been applied the observation that the probability of walking from i to j then coming back to i is always the same as starting from j, reaching i and then coming back to j. Thus the transformation function is defined as:

$$Rel(i \mid j) = Rel(j \mid i) = \frac{P^{(t)}(j \mid i) + P^{(t)}(i \mid j)}{2} \tag{3.2}$$

and the score are normalized to range {0, 1} using:

$$Rel(i \mid j) = \frac{Rel(j \mid i)}{\max Rel(j \mid i)} \tag{3.3}$$

Named Entity Disambiguation step

The final step of the methodology consists of choosing a single meaning (*concept*) for each *entity surface*, exploiting *Semantic Relatedness* scores derived by the graph. Given $S = \{s_1, \ldots, s_n\}$ the set of *surfaces* in a document, $C = \{c_{1_k}, \ldots, c_{m_k}\}$ (with $k = 1 \cdots \mid S \mid$), the set of all their possible senses (*concepts*) and R the matrix of relatedness $Rel(i \mid j)$ with each cell indicating the strength of relatedness between concept c_{i_k} and concept $c_{j_{k'}}$ (where $k \neq k'$, that is c_{i_k} and $c_{j_{k'}}$ have different surface forms), the *entity disambiguation algorithm* is defined as a function $f : S \rightarrow C$, that given a set of *surfaces* S returns the list of disambiguated

concepts, using the concept relatedness matrix R. Different kind of such functions f have been designed and results are compared in Chapter 4.

As first and simple disambiguation function is the ***highest method***: $cand_{k_i}$, the list of candidates winner concepts for each surface, is built, with i being the candidate concept for *surface* k ($k = 1 \cdots \mid S \mid$); if some of *surfaces* k has more that one candidate winner, for each k *surface* with multiple i values, the *concept* that among the candidates has the highest value in the matrix R is picked.

The ***combination method*** calculates for each concept c_{i_k} the sum of relatedness with all different concepts $c_{j_{k'}}$ from different surfaces (such as $j \neq i$, $k' \neq k$). Given $V = \{v_1, \ldots, v_{|C|}\}$ the vector of such values, the function returns for each surface s_k the concept c_{i_k} that has the max v_i.

The ***propagation method*** works as follows: taking as seed the highest similarity value in the matrix R, the 2 concepts i and j giving that value are fixed: for their surface form k and k' rows and columns in the matrix R coming from other concepts for the same surfaces (all c_{t_k} and $c_{t_{k'}}$ with $t \neq i$ and $t \neq j$) are deleted. This step is repeated recursively, picking the next highest value in R. The stop condition consists of having one concept row in the matrix R for each surface form.

Chapter 4 reports evaluations of SRaNED over two different datasets.

3.3 NLP framework: META (Multi-languagE Text Analyzer)

In order to implement and evaluate NED algorithms a Natural Language Processing platform that provides tools for the automatic elaboration of texts is required. Moreover this platform must be able to work with several languages because another goal of this thesis is to use NED not only for English but also for other languages and in particular for Italian. Furthermore, Natural Language Processing *(NLP)* has significant impact on many relevant Web-based and Semantic Web applications, such as information retrieval and information extraction. Tools that are able to make easy the development of NLP applications are very useful. META *(MultilanguagE Text Analyzer)* [Basile et al. (2008d)] is a tool for text analysis which implements some NLP functionalities.

META is an infrastructure for language processing that contributes along several dimensions:

- The system is designed to separate cleanly low-level tasks, such as data storage, location and loading of components and execution of processes from the data structures and algorithms that actually process human language.
- Providing a baseline set of language processing components that can be extended and/or replaced by users as required.
- Make easy the development of NLP algorithms over different languages and the integration of WSD.
- Allows the integration with Information Retrieval and Information Extraction external tool.

META has been employed for the processing collection of documents in different scenarios:

- disambiguation of a whole collection of documents in English [Basile et al. (2007c)];
- disambiguation of a whole collection of documents in Italian [Basile et al. (2007b)];

- indexing of a whole collection of scientific papers and movie descriptions for personalized filtering [Semeraro et al. (2007)];
- indexing of documents in an Electronic Performance Support System (EPSS) [Basile et al. (2007a)];
- within Lexical Browser, a service that, given a lexeme (an abstract unit of morphological analysis in linguistics, which roughly corresponds to a set of words that are different forms of the same word), returns all syntactic and semantic information collected from a list of lexical and semantic resources [Gentile et al. (2008)].

3.3.1 System Architecture

META is a tool for text analysis which implements some NLP functionalities; in addition it provides the tools for semantic indexing of documents. META is composed by three main components:

1. *collection manager:* this component provides the tools for the importation of text from several formats (HTML, PDF, DOC,...) and the organization of documents in collections. Also this component allows the segmentation of document (e.g. title, abstract, ...) and the annotation of this section with tags stored in a domain ontology;
2. *NLP engine:* this component implements the engine able to execute the different NLP annotators. A NLP annotator is a component that performs a particular NLP step (e.g. tokenization, stop word elimination, pos-tagging, ...). The NLP engine schedules these components, loads the needed lexical resources for each annotator and runs the annotator over all the documents into a collection.
3. *export manager:* this component is able to export the results produced by the NLP engine into a new format, for example: databases, XML, RDF, inverted index and so on. This component can export a whole collection or a single document.

Figure 3.2 sketches the interaction between these components.

The *Collection Manager* imports the text from different sources and organizes it in documents. The output of this process is a collection of documents. A document is represented into the system by an ID, the raw text extracted by the original source and an ID_LANG that identifies

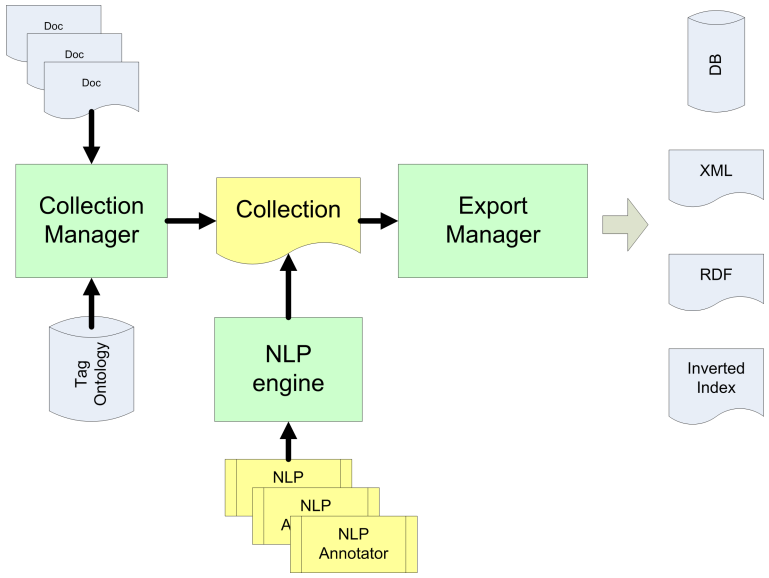

Figure 3.2: META conceptual architecture

the document language. Furthermore, it is possible to associate some information to the whole collection and to a subset of documents using tags stored in a domain ontology. Now the documents are ready to be processed.

First, the NLP engine autodetects the document language[1], this step is necessary in order to load the lexical resources that can be different for each languages. Second, the NLP engine normalizes the text: the original text is modified to prepare it for the following steps (for example, all formatting characters are removed). Third, the engine tokenizes the text: split up normalized text into tokens. After these steps each document has a list of tokens and it is possible to associate a set of annotations to each token . An annotation is a pair *(annotation_name,value)*. The annotations are produced out by each NLP annotator. Indeed, after the initial steps, the engine schedules the NLP annotators and run them.

Currently, the following annotators have been developed :

1. Stop words elimination: all commonly used words are deleted;
2. Stemming: the process of reducing inflected (or sometimes derived)

[1]The language is detected automatically but the user can set a default language for each document.

words to their stem. The *Snowball stemmer*[2] has been adopted;

3. POS-tagging: the process of assigning a part-of-speech tag to each token. A JAVA version of *ACOPOST tagger*[3] using Trigram Tagger T3 algorithm has been developed. It is based on Hidden Markov Models, in which the states are tag pairs that emit words;
4. Lemmatization: the process of determining the lemma for a given word. WordNet Default Morphological Processor (included in the WordNet distribution) for English has been used. For the Italian language, a different lemmatizer that exploits the *Morph-it!* morphological resource[4] has been built;
5. NER driven by Ontologies: the process of finding ontology instances into the text;
6. NER using Support Vector Machine[5] trained exploiting CoNLL 2002 Shared Task[6] for English and I-CAB Evalita 2007[7] for Italian;
7. WSD: the problem of selecting a sense for a word from a set of predefined possibilities exploiting a sense inventory that usually comes from an electronic dictionary or thesaurus.
8. NED: using the WibNED algorithm proposed in this thesis.

After the NLP engine has ran the pipeline, its output could be exported in several formats using the export manager. This component is developed to transform the internal output of META in a formal format such as XML, RDF, inverted index and so on.

3.3.2 Document representation

The internal representation of META is a collection that contains a list of documents. Each document is composed by at least one segment. A segment contains a list of tokens and each token has at least one annotation. An annotation represents a particular feature extracted during the text processing (e.g. stemming, lemma, entity, sense, ...). Figure 3.3 shows the conceptual structure of the document.

[2] http://snowball.tartarus.org/
[3] http://acopost.sourceforge.net/
[4] http://sslmitdev-online.sslmit.unibo.it/linguistics/morph-it.php
[5] http://chasen.org/~taku/software/yamcha/
[6] http://www.cnts.ua.ac.be/conll2002/ner/
[7] http://evalita.fbk.eu/2007/tasks/ner.html

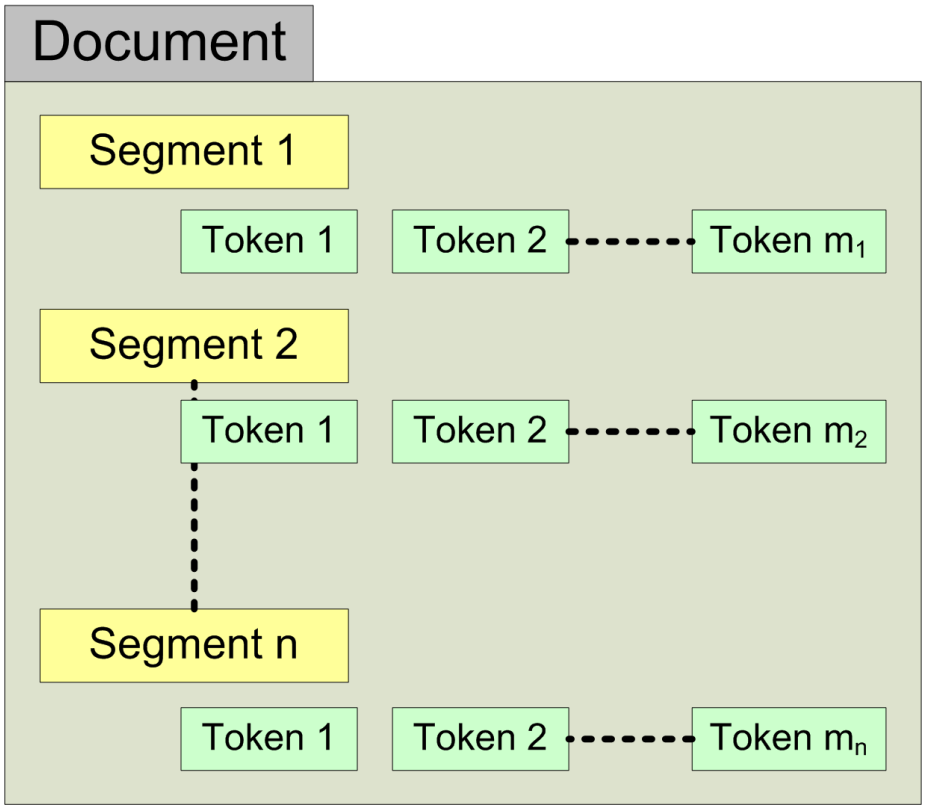

Figure 3.3: Conceptual document structure in META

For example, if the user wants to analyze the text '*In this paper we present META (MultilanguagE Text Analyzer), it is a tool for text analysis which implements some NLP functionalities.*', the NLP engine executes the following tasks: tokenization, stemming, pos-tagging, lemmatization and WSD. The output of the system is a list of tokens and each extracted feature is an annotation. Figure 3.4 shows the conceptual token structure for the token *paper*. In figure 3.4 the token "paper" has a *sense annotation* and the value is *n12660433* that identifies the WordNet synset assigned by the WSD algorithm.

When the token represents a NE has also an *entity annotation* and the value is an URI from Wikipedia, in the case of using Wikipedia as *entity inventory*, but could be for example an ontology URI if the *entity*

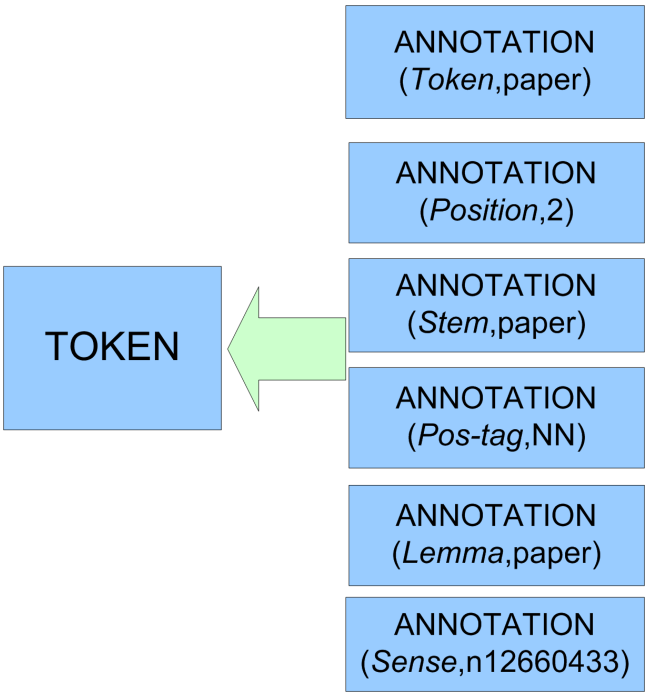

Figure 3.4: Conceptual token structure in META

inventory is in the form of an ontology.

Chapter 4

Evaluation

NED algorithms proposed in Chapter 3 have been tested over different datasets.
For english language two gold standard datasets for the task of NED has been proposed by Cucerzan (2007). Both datasets are publicly available[1].
For italian language there is no official dataset for the NED task (neither for training nor for testing). For this reason it has been necessary to build an *ad hoc* dataset.
This Chapter presents english datasets in Section 4.1.1 and italian dataset in Section 4.1.2, which also describes the automatic procedure to build it.

4.1 Experimental Datasets for NED

4.1.1 NED Datasets for English

NEWS dataset The first dataset used for experiments, that will be referred after as *NEWS*, consists 20 news stories: for each story it is provided the list of all entities, annotated with the corresponding page in Wikipedia. The number of entities in each story can vary form 10 to 50. Some of the entities have a blank annotation, because they do not have a corresponding page in the Wikipedia collection: among all the identified entities, 370 are significantly annotated in the test data. The whole dataset is a single text file where the beginning and the end of

[1] http://research.microsoft.com/users/silviu/WebAssistant/TestData

each document (a news story) is marked respectively with the symbols *~~~BOD* and *### EOD*.

For each document there is the list of appearing entities, one for each line. An example of document is the following:

```
~~~ BOD
Timberlake                  Justin Timberlake
Diaz                        Cameron Diaz
N' Sync                     'N Sync
Justin Timberlake           Justin Timberlake
Cameron Diaz                Cameron Diaz
Star magazine               Star (magazine)
Star                        Star (magazine)
Diaz                        Cameron Diaz
Christmas                   Christmas
Vail                        Vail, Colorado
Colo.                       Colorado
Timberlake                  Justin Timberlake
Memphis                     Memphis, Tennessee
N' Sync                     'N Sync
Saturday Night Live         Saturday Night Live
Diaz                        Cameron Diaz
Timberlake                  Justin Timberlake
Kids' Choice Awards         Nickelodeon Kids' Choice Awards
Timberlake                  Justin Timberlake
Veronica Finn               Veronica Finn
Britney Spears              Britney Spears
Spears                      Britney Spears
Innosense                   Innosense (band)
Lou Pearlman                Lou Pearlman
### EOD
```

As input for the system the list of entities spotted in the benchmark data has been used. For each entity the list of all possible meaning is retrieved, e.g. for surface "Alabama" following concepts are retrieved:

Alabama $\longrightarrow$ [$AlabamaClaims$ | $Genus$ | $CSSAlabama$ | $AlabamaRiver$ | $Alabama(people)$ | $Noctuidae$ | $Harvest(album)$ | $USSAlabama$ | $Alabamalanguage$ | $Alabama(band)$ | $Moth$ | $UniversityofAlabama$ | $Alabama, NewYork$]

WIKI dataset The second dataset, that will be referred after as *WIKI*, consists of 350 Wikipedia entity pages selected randomly. The text articles contain a total of 5,812 entity surface forms. We performed our evaluation on 3,136 entities, discarding all non-recallable surfaces, that is all those surfaces having no correspondance in the Wikipedia collection. Indeed, an error analysis carried out by Cucerzan on this dataset showed that it contains many surface forms with erroneous or out-of-date links, as reported in [Cucerzan (2007)]. The structure of this dataset is the same already descripted for the NEWS dataset.

4.1.2 A NED Dataset for Italian

The evaluation of a NED algorithm which gives as output Wikipedia URIs needs a dataset containing ambigous entites, already tagged with the correct URI belonging to Wikipedia. Such a resource was not available for italian language, for this reason it has been constructed a dataset complying with such requests. The builded dataset consists of 752 documents extracted from italian Wikipedia.

It has been implemented a procedure to automatic build the annotated corpus, starting from a list of ambigous entities (i.e. entities whose surface form has an associated *Disambiguation Page*[2] in Wikipedia).

A list of 100 ambiguos surface form has been used, taken from italian Wikipedia, $A = (a_1, \ldots, a_{100})$. For each a_i the related *Disambiguation Page* on Wikipedia has been accessed and the most significative senses for a_i, $s_{a_i} = (s_1, \ldots, s_j)$, with j ≤ 4 have been picked, considering only senses referring to Named Entities and using heuristics to reject poor senses. For example, considering the *disambiguation page* for the italian word "mosca", the sense referring to *Mosca (Moskva), the capital and the largest city of Russia*, has been stored whereas the sense referring to *Muscomorpha, a group of flies*, has been ignored because it is a common noun word. Starting from $S = (s_{a_1}, \ldots, s_{a_t}, \ldots, s_{a_{100}})$ for each s_j in s_{a_t},

[2] http://en.wikipedia.org/wiki/Category:Disambiguation_pages

up to 5 generic Wikipedia articles have been chosen, with the specific that they contain at least a link to the sense s_j. Each article has been processed using META [Basile et al. (2008d)] and has been stored as a single file, using a IOB like format, similar to CoNLL 2003 Named Entity Recognition corpus[3]: each row contains a token, its Part Of Speech tag, its lemma and a final tag which is valued as O if the token has not been recognized as an entity, B-<Wikipedia URI for the entity> if the token is the beginning of an entity and I-<Wikipedia URI for the entity> if the token continues an entity.

Each text contains on the average 89 entities. Table 4.1 reports a piece of a document to show an example of text. It is a document included in the corpus, specifically it is an article taken from italian Wikipedia about *The Beastie Boys*. In this piece of text there is only one entity, represented by the two words John Berry. Together with entity annotations, the corpus also contains the Part Of Speech and the stem for each word, thus allowing to refine and improve computation over the corpus.

4.2 WibNED Experiments

The experiment has been performed following the methods generally used to evaluate WSD algorithms. The WSD task is considered not an end in itself but rather an "intermediate task" which contributes to an overall task such as information retrieval, ontology building, etc. This opens the possibility of two types of evaluation for WSD work (using terminology borrowed from biology): in vitro evaluation, where WSD systems are tested independently of any application, using specially constructed benchmarks; and evaluation in vivo, where, rather then being evaluated in isolation, results are evaluated in terms of their contribution to the overall performance of a system designed for a particular application (e.g., information retrieval). In this instance in vitro evaluation has been adopted in order to evaluate the accuracy and the potentialities of the algorithm in an independent way. In vitro evaluation, despite its artificiality, enables close examination of the problems plaguing a given task. In its most basic form this type of evaluation involves comparison of the output of

[3] http://www.cnts.ua.ac.be/conll2003/ner/

Table 4.1: Corpus example

Il	RS	Il	O
nome	SS	nome	O
"	XPO	"	O
Beastie	SP	beastie	O
"	XPO	"	O
,	XPW	,	O
inventato	VSP	inventare	O
dall	SN	dall	O
’	XP	’	O
ex	SN	ex	O
componente	SS	componente	O
John	SPN	John	B-http:it.wikipedia.org/wiki/John_Berry
Berry	SPN	Berry	I-http:it.wikipedia.org/wiki/John_Berry
,	XPW	,	O
é	VI	essere	O
l	SN	l	O
’	XP	’	O
acronimo	AS	acronimo	O
della	ES	della	O
frase	SS	frase	O
"	XPO	"	O
Boys	YF	Boys	O
Entering	YF	Entering	O
Anarchistic	YF	Anarchistic	O
States	YF	States	O
Towards	YF	Towards	O
Inner	SPN	Inner	O
Excellence	SPN	Excellence	O
"	XPO	"	O

a system for a given input, using measures such as precision and recall. Alternatively, in vitro evaluation can focus on study of the behavior and performance of systems on a series of test suites representing the range of linguistic problems likely to arise in attempting WSD. Considerably deeper understanding of the factors involved in the disambiguation task is required before appropriate test suites for typological evaluation of WSD results can be devised. The in vitro evaluation demands the creation of a manually sense-tagged reference corpus containing an agreed-upon set of sense distinctions. The idea is to build a corpus, find the entity into the corpus and annotate the entity with relative sense. In this experiment Wikipedia is used as *sense inventory* for the entities because it's the same used by the WibNED algorithm.

WibNED is implemented in JAVA, by using Lexical Collector web service [Gentile et al. (2008)] as accessing point to the Wikipedia *sense inventory*. The WibNED algorithm has been used over the dataset described in Section 4.1.2 in order to evaluate its effectiveness. The metric that has been used to evaluate results is *accuracy*, following previous work experiments [Cucerzan (2007)]. Accuracy describes the ratio between entities correctly labelled by WibNED and total number of entities within the document.

Experimetal results are showed in figure 4.1: on the x-axis are reported single documents while on the y-axis is reported the accuracy of WiBNED for each document. Documents on x-axis are ordered according to ascending accuracy.

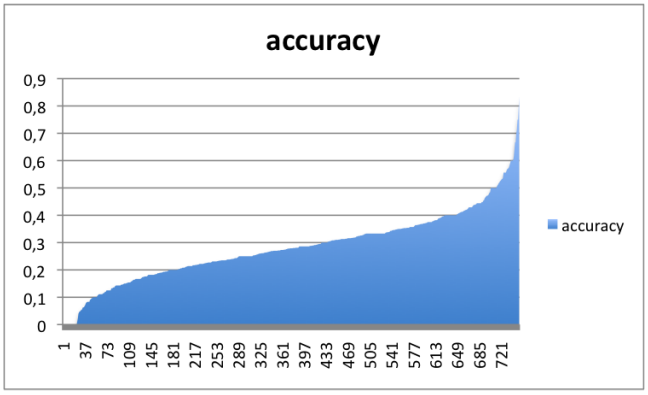

Figure 4.1: Accuracy of WiBNED on 752 documents

Total Number of Entities	67029
Total Number of correctly labelled Entities	19131
Average number of Entities per Document	89
Average number of Correctly labelled Entities per Document	25
Total Accuracy	0.285
Average Accuracy on single Document	0,282
Minimal Accuracy	0,000
Maximal Accuracy	0,833

TABLE 4.2: WibNED Results

Statistics are reported in table 4.2. The table provides the total number of entities within the corpus, the total number of correctly labelled entities by WiBNED algorithm, the average number of entities on single documents, the average number of correctly labelled entities per document. Results about accuracy are showed: total accuracy which is calculated considering the total number of entities and total number of correctly labelled entities, regardless of distribution between documents, average accuracy considering the mean of accuracy of all documents. The minimal and maximal values reported for accuracy are listed in the table.

The results are not suprising if we consider that WibNED is a very simple algorithm based only on the string-matching between the words in Wikipedia definition and the words within the context of the target entity. The algorithm achieves on average 28,2% of accuracy. Taking into account more informative features borrowed from Wikipedia, such as category labels associated to each article or internal links, could improve results of the algorithm. However, the choice has been to completely change methodology. Next Section shows more encouraging results obtained with the SRaNED algorithm.

4.3 SRaNED Experiments

The experiment has been performed with the same methodology as previous one, using an "in vitro evaluation". The goal is to prove that the usage of Semantic Relatedness scores is profitable for the issue of NED

and that the graph of interconnections between entities is influent for the disambiguation decision. As benchmark to test the SRaNED algorithm, the NEWS and the WIKI datasets described in Section 4.1.1 have been used.

As described in Chapter 3 for each surface are retrieved the possible concepts. All identified possible concepts for each surface are used to build a graph. After running the Random Walk on the built graph and transforming the transition matrix in a relatedness matrix, an upper triangular matrix is obtained, with a score of relatedness between different concepts, belonging to different surfaces. The *weight model* (*wm*), that is weights associated to each type of edges in the graph, has been determined applying a simulated annealing method [Nie et al. (2005)]. The algorithm explores the search space of all possible combinations of feature weights and iteratively reduces the difference between a gold standard solution and that of our system. The algorithm allows to run the method on one dataset in an iterative manner, where in each iteration, the algorithm generates a random *wm* for the feature set and scores our system results obtained with that model against the gold standard. If a *wm* obtained in the current iteration produces better results than the previous, the simulated annealing algorithm will attempt to adjust weights based on that model in next iterations. Thus, by running simulated annealing for relatively large number of iterations, we expect the system performance to converge; by which we obtain the final optimum weight model for that feature set. This tuning has been done in advance using a standard testing dataset for semantic relatedness, the WordSimilarity-353 Test Collection [Finkelstein et al. (2002)], and empirically derived the optimum weight model for the chosen feature set.

Performance have been evaluated in terms of accuracy, the same as for the WibNED experiment.

Results obtained applying all defined disambiguation functions to the relatedness matrix are shown in table 4.3 for the *NEWS* dataset and in table 4.4 for the *WIKI* dataset. Both tables also report figures obtained by Cucerzan (2007) on the same datasets .

In the first experiment the best result equals the best available system at the state of the art, with an accuracy of 91.4 % and all proposed functions are above the baseline of 51.7% (baseline returns always the

first available result). Between three proposed methods, the *combination method* obtained the best result, equalling the best available system at the state of the art. The *highest method* achieves results below the state of the art of 91.4%, even if, with an accuracy of 82.21% is far over the baseline of 51.7% (baseline returns always the first available result). The motivation can be that it takes into account only the best relatedness score for each concept to decide sense assignment, without considering the rest of the scores. The *propagation method* works even worse because adds to the disadvantage of the first one also the propagation of errors. It reaches an accuracy of 68.68%, which is in the middle between the baseline and the state of the art.

TABLE 4.3: Comparison of proposed Named Entity Disambiguation Functions on *NEWS* dataset

Literature Systems	Accuracy	Function	Accuracy
Cucerzan baseline [Cucerzan (2007)]	51.7%	Highest	82.21%
Cucerzan [Cucerzan (2007)]	**91.4%**	**Combination**	**91.46%**
		Propagation	68.68%

To consolidate this result the same experiment has been conducted over the *WIKI* dataset.

TABLE 4.4: Comparison of proposed Named Entity Disambiguation Functions on *WIKI* dataset

Literature Systems	Accuracy	Function	Accuracy
Cucerzan baseline [Cucerzan (2007)]	86.2%	Highest	87.12%
Cucerzan [Cucerzan (2007)]	**88.3%**	**Combination**	**89.83%**
		Propagation	84.34%

The second experiment definitively confirms the trend reported in the first one. The three proposed disambiguation functions have the same behavior on the *WIKI* dataset. The *combination method* is the best one: it achieves an accuracy of 89.83%, outperforming the accuracy of 88.3% reported by the state-of-the art system. The *highest method* is between the baseline and the state-of-the-art system, with an accuracy of 87.12%. The *propagation method* is the worst one: with an accuracy of 84.34 % is under the baseline of 86.2%.

As expected and already assessed in the *NEWS* experiment, the *combination method* performs much better than others, outperforming the state of the art system on the *WIKI* dataset. The motivation can be found in the fact that it considers relatedness scores in their entirety, giving value to the interaction of all concepts instead of couples of concepts. Such value can be considered an encouraging outcome for the proposed novel method: the second experiment reinforces results of the first one, and states the correctness of the proposed methodology.

Chapter 5

Applications

This chapter takes into account two different scenarios where NLP techniques and specifically NE management have been used.
Section 5.1 describes SENSE, an Information Retrieval (IR) system which introduces *semantic levels* to overcome the limitations of traditional IR systems. NEs constitute one of the *semantic levels* of the system [Basile et al. (2008a,c,b)].
Section 5.2 describes an Electronic Performance Support System (EPSS), which benefits of NE management facilities [Iaquinta et al. (2007); Basile et al. (2007a,d)].

5.1 Semantic Search

The traditional strategy performed by IR systems is ranked keyword search: For a given query, a list of documents, ordered by *relevance*, is returned. Relevance computation is primarily driven by a basic string-matching operation. To date, several attempts have been made to deviate from the traditional keyword search paradigm, often by introducing some techniques to capture word meanings in documents and queries. The general feeling is that dealing explicitly with *only* semantic information does not improve significantly the performance of text retrieval systems. Information resulting from WSD and NED processes have been exploited to build a semantic representation of documents to use within SENSE (SE-

mantic N-levels Search Engine), an IR system that tries to overcome the limitations of the ranked keyword approach, by introducing *semantic levels* which integrate (and not simply replace) the lexical level represented by keywords. Semantic levels provide information about word meanings, as described in a reference dictionary, and named entities. The SENSE IR system manages documents indexed at three separate levels, keywords, word meanings, and entities; the system also combines keyword search with semantic information provided by the two other indexing levels.

Ranked keyword search is quite successful, in spite of its obvious limits basically due to polysemy and synonymy. The result is that, due to synonymy, relevant documents can be missed if they do not contain the exact query keywords, while, due to polysemy, wrong documents could be deemed as relevant. These problems call for alternative methods that work not only at the lexical level of the documents, but also at the *meaning* level.

Any attempt to work at the meaning level must solve the problem that, while words occur in a document, meanings do not, since they are often hidden behind words. For example, for the query "apple", some users may be interested in documents dealing with "apple" as a fruit, while other users may want documents related to the company. Some linguistic processing is needed in order to provide a more powerful "interpretation" both of the user needs behind the query and of the words in the document collection. This linguistic processing may result in the production of *semantic information* that provide machine readable insights into the meaning of the content. As shown by the previous example, named entities mentioned in the documents constitute important part of their semantics. Therefore, semantic information could be captured from a text by looking at *word meanings*, as they are described in a reference dictionary (e.g. WordNet [Miller (1995)]), and *named entities*.

SENSE IR system manages documents indexed at multiple separate levels: keywords, senses (word meanings), and entities. The system is able to combine keyword search with semantic information provided by the two other indexing levels. In particular, for each level:

- a *local scoring function* weighs elements belonging to that level according to their informative power;

- a *local similarity function* computes document relevance by exploiting the above-mentioned scores.

Finally, a *global ranking function* is defined in order to combine document relevance computed at each level.

5.1.1 N-Levels model

In the vector model for IR [Salton and McGill (1983)], documents and queries are represented in a t-dimensional vectorial space and tf-idf schemes are used to weigh index terms. Term weights are used to compute the *degree of similarity* between each document d in the collection and the user query q, according to a ranking function $R(q, d)$, defining an ordering among the documents with respect to the query.

The main idea underlying the definition of an open framework to model different semantic aspects (or levels) pertaining document content is that there are several ways to describe the semantics of a document. Each semantic facet needs specific techniques and ad-hoc similarity functions. In the SENSE IR framework a different IR model is defined for each level in the document representation. Each level corresponds to a *logical view* that aims at describing one of the possible semantic spaces in which documents can be represented. The adoption of different levels is intended to guarantee acceptable system performance even when not all semantics representations are available for a document.

The keyword level is supposed to be always present and, when also other levels are available, these ones are used to offer enhanced retrieval capabilities. Furthermore the framework allows to associate each level with the appropriate representation and similarity measure. The following semantic levels are currently available in the framework:

Keyword level - the entry level in which the document is represented by the words occurring in the text.

Word meaning level - this level is represented through *synsets* obtained by WordNet, a semantic lexicon for the English language. A synset is a set of synonym words (with the same meaning). Word Sense Disambiguation algorithms are adopted to assign synsets to words.

Named Entity level - this level consists of entities recognized into the document text and tagged with a unique Wikipedia URI. A NED algorithm is adopted to assign a Wikipedia URI to words that represent entities.

For example, it is possible to associate each named entity with a weight which takes into account the presence of other related entities found into the document. The user can exploit this level for different tasks:

1. to browse a domain ontology looking for specific entities or properties;
2. to perform a keyword search and, if some term in the query is recognized as a named entity, documents containing such entity are highlighted on the hit of the returned document list.

Analogously, N different levels of representation are needed for representing queries. The N query levels are not necessarily extracted simultaneously from the original keyword query issued by the user: A query level can be obtained when needed. For example, the ranked list of documents for the query "Apple growth" might contain documents related to both the growing of computer sales by Apple Inc. and the growth stages of apple trees. Then, when the system will collect the user feedback (for instance, a click on a document in which "Apple" has been recognized as a named entity), the query representation for the named entity level is produced.

The notion of relevance $R(q, d)$, which computes the *degree of similarity* between each document d in the collection and the user query q, needs also to be extended. The relevance must be evaluated at each level by defining a proper *local similarity function* that computes document relevance according to the weights defined by the corresponding local scoring function. Since the ultimate goal is to obtain a *single* list of documents ranked in decreasing order of relevance, a *global ranking function* is needed to merge all the result lists that come from each level. This function is independent of both the number of levels and the specific local scoring and similarity functions because it takes as input N ranked lists of documents and produces a unique merged list of most relevant documents. Section 5.1.5 describes the adopted global ranking function.

5.1.2 SENSE System Architecture

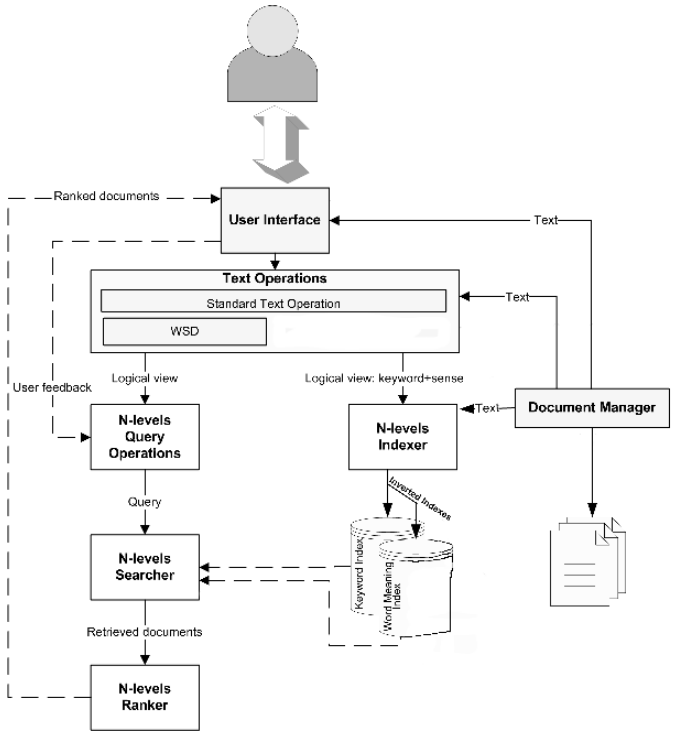

Figure 5.1: SENSE System Architecture

SENSE is a semantic IR system based on the N-Levels model described in the previous section. Figure 5.1 depicts the system architecture and shows the modules involved in the information extraction and retrieval processes. In more detail:

- Document Manager - It manages document collections to be indexed. It is invoked by the User Interface module to display the results of a user query.
- Text Operations - It performs basic and more advanced NLP operations. Basic operations implemented are: *Stop words elimination*, *Stemming* (the *Snowball* stemmer is adopted[1]), *POS-tagging*, *Lemmatization* and Named Entity Recognition (NER). For POS-

[1] http://snowball.tartarus.org/

tagging, a JAVA version of *ACOPOST tagger*[2] has been implemented; it adopts Trigram Tagger T3 algorithm based on Hidden Markov Models. For lemmatization, the WordNet Default Morphological Processor, as is included in the WordNet 2.0 distribution for English, has been used. The NER task has been performed with Yamcha [Kudo and Matsumoto (2003)]. Besides basic NLP processing, more advanced procedures were designed for the semantic levels of SENSE: *Named Entity Disambiguation (NED)* and *Word Sense Disambiguation (WSD)*. The core component that performs all the steps (WSD and NED included) needed for building the document representation at the meaning level is META [Basile et al. (2008d)], already described in Chapter 3.

- User Interface - It provides the query interface, which is not just a textbox where keywords can be typed since it allows users to issue queries involving semantic levels.

The core of the N-Levels indexing and retrieval processes consists of the following modules:

- N-Levels Indexer - It creates and manages as many inverted indexes as the number of levels into the N-levels model. While the Text Operations component provides the features corresponding to the different levels, the N-Levels Indexer computes the local scoring functions defined for assigning weights to features.
- N-Levels Query Operations - It reformulates user needs so that the query can be executed over the appropriate inverted indexes.
- N-Levels Searcher - It retrieves the set of documents matching the query, for each level identified by Text Operations. It implements the local similarity functions defined in the model.
- N-Levels Ranker - It arranges documents retrieved by the Searcher into a unique list to be shown to the user. For each level involved into the search task, it ranks documents according to the local similarity function and then merges all the local lists into a single list by using the global ranking function.

[2] http://acopost.sourceforge.net/

The core components that perform the N-Levels indexing and retrieval processes are implemented on the LUCENE API[3]. LUCENE is a full-featured text search engine library that implements the vector space model. An extension of the LUCENE API has been implemented, the N-LEVELS LUCENE CORE, to meet all the requirements of the proposed model.

5.1.3 Meaning Level

In SENSE, features at the meaning level are *synsets* obtained from WORDNET 2.0. It groups English words into sets of synonyms called *synsets*, provides short general definitions (*glosses*), and records various semantic relations between synonym sets. WORDNET distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. Each synset is assigned with a unique identifier and contains a set of synonymous words or collocations; different senses of a word occurs in different synsets.

In order to assign synsets to words, a WSD strategy has been adopted. The WSD algorithm adopted in SENSE is an improved version of JIGSAW algorithm [Basile et al. (2007b)]. The basic idea of the algorithm is to combine three different strategies to disambiguate nouns, verbs, adjectives and adverbs respectively. The main motivation behind our approach is that the effectiveness of a WSD algorithm is strongly influenced by the Part of Speech (POS) tag of the target word.

The idea behind the adoption of WSD is that each document is represented at the meaning level by the senses conveyed by the words, together with their respective occurrences. The WSD procedure produces a synset-based vector space representation, called bag-of-synsets (BOS). In this model a document is represented by a synset vector, rather than a word vector.

Let D be a collection of M documents. The j-th document in D is represented as:

$$d_j = \langle t_{j1}, t_{j2}, \ldots, t_{jn} \rangle,\ j = 1, \ldots, M$$

where t_{jk} is the k-th synset in d_j, n is the total number of synsets in

[3]http://lucene.apache.org/

d_j. Document d_j is represented in a $|V|$-dimensional space by a synset-frequency vector, V being the vocabulary for D (the set of distinct synsets recognized by the WSD procedure in the collection):

$$f_j = \langle w_{j1}, w_{j2}, \ldots, w_{j|V|} \rangle,\ j = 1, \ldots, M$$

where w_{jk} is the weight of the synset t_k in d_j, computed according to the local scoring function defined in the next section.

Synset Scoring Function Given a document d_i and its synset representation $X = [s_1, s_2, \ldots, s_k]$, the idea is to compute a *partial* weight for each $s_j \in X$, and then to improve this weight by finding out some relations between synsets belonging to X. The partial weight, called SFIDF (synset frequency, inverse document frequency), is computed according to a strategy resembling the tf-idf score for words:

$$\text{SFIDF}(s_j, d_i) = \underbrace{\text{TF}(s_j, d_i)}_{\text{synset frequency}} \cdot \underbrace{log\frac{|\ C\ |}{n_j}}_{\text{IDF}} \tag{5.1}$$

where $|\ C\ |$ is the total number of documents in the collection and n_j is the number of documents containing the synset s_j. $\text{TF}(s_j, d_i)$ computes the frequency of s_j in document d_i.

The local scoring function for synsets relies on *"Semantic Domains"*, which are areas of human discussion, such as POLITICS, ECONOMY, SPORT, which exhibit their own terminology and lexical coherence. WordNet Domains [Magnini and Cavaglià (2000)] has been adopted. It is an extension of WordNet in which each synset is annotated with *one or more* domain labels[4]. The domain set of WordNet Domains is composed of about 200 domain labels. The idea of including WordNet Domains in the synset scoring function is based on the *lexical coherence* assumption, claiming that a great percentage of concepts expressed in the same document belongs to the same domain. The availability of WordNet Domains makes it possible to give more weight to synsets belonging to more relevant domains in d. The main advantage of this approach is that WSD errors can be mitigated by domain information. For example, if the noun "bank" was incorrectly disambiguated as "sloping land" (domain: GEOGRAPHY), while its correct sense was "financial institution"

[4]Freely available for research at `http://wndomains.itc.it`

(domain: ECONOMY), this error could be recovered by observing that ECONOMY was a common domain in d, while GEOGRAPHY was very rare.

Two different kinds of domain relevance have been taken into account: The relevance of a domain with respect to a specific synset, and the relevance of a domain with respect to the whole set of synsets recognized in a document. In the following, two functions that estimate both kinds of relevance, called *domain relevance* and *document domain relevance*, respectively, are defined. Let $D = \{D_1, D_2, \ldots, D_m\}$ be the set of WORDNET DOMAINS. Intuitively, a domain $D_j \in D$ is relevant for a specific synset s if D_j is relevant for the texts in which s usually appears. As an approximation, the information in WORDNET DOMAINS can be used to estimate such a function. Let $Dom_j = \{D_{j1}, D_{j2}, \ldots, D_{jh}\}$, $D_j \subseteq D$, be the set of domain labels assigned to synset s_j in WORDNET DOMAINS. The domain relevance function is defined as:

$$Rel(D_i, s_j) = \begin{cases} 1/\mid Dom_j \mid & \text{if } D_i \in Dom_j \\ 1/m & \text{if } Dom_j = \text{FACTOTUM} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where $m = \mid D \mid$. The domain FACTOTUM covers generic synsets not belonging to a specific domain (they correspond to general language and may appear in any context). Under these settings, generic synsets (FACTOTUM) have low relevance values for each domain, while domain-oriented synsets have high relevance values for a specific domain.
$Rel(D_i, s_j)$ can be perceived as an estimated prior probability of the domain given the synset. Given a document d and its synset representation $X = [s_1, s_2, \ldots, s_k]$, the relevance of domain D_i in d is defined as the percentage of synsets in X assigned to D_i. Formally:

$$DocRel(D_i, X) = \begin{cases} \#(s_j, D_i)/\mid X \mid & \text{if } D_i \in Dom_j \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where $\#(s_j, D_i)$ is the number of $s_j \in X$ for which $D_i \in Dom_j$. For each s_j, the relevance of the domains assigned to s_j is encapsulated into a domain factor α:

$$\alpha = \sum_{D_{jh} \in Dom_j} Rel(D_{jh}, s_j) \cdot DocRel(D_{jh}, X) \tag{5.4}$$

The domain factor is then exploited to compute the final local score for synset s_j in d_i as $\text{SFIDF}(s_j, d_i) \cdot (1 + \alpha)$.

Synset Similarity Function The local similarity functions for both the meaning and the keyword levels are computed using a modified version of the LUCENE default document score. Given a query q and a document d_i, the synset similarity is computed as:

$$synsim(q, d_i) = C(q, d_i) \cdot \sum_{s_j \in q} (\text{SFIDF}(s_j, d_i)(1 + \alpha) \cdot N(d_i)) \tag{5.5}$$

where: $\text{SFIDF}(s_j, d_i)$ and α are computed as described in the previous section, $C(q, d_i)$ is the number of query terms in d_i, $N(d_i)$ is a factor that takes into account document length normalization.

5.1.4 Named Entity Level

The NED algorithm used in this work is WibNED [Gentile et al. (2009)]. Each token that refers to an Entity in the original document is tagged with the Wikipedia URI that better represents the Entity meaning.

Considering that the words to disambiguate for Named Entity Disambiguation are only those representing an Entity, the algorithm works as follows.

Similarly to the meaning level, documents are represented at the entity level by using an adaptation of the vector space model, the rapresentation adopted for this level is a *bag-of-entities* rather than a *bag-of-synsets*. The vocabulary is the set of entities recognized by the NER text operation in the collection; specifically each entity is identified by the URI of the entity instance (borrowed from Wikipedia). As first attempt, a classical tf-idf heuristic has been adopted to score entities and cosine similarity has been implemented as local similarity function.

5.1.5 Global Ranking

The strategy for defining the *global ranking function* is inspired by prior work on meta-search engines [Farah and Vanderpooten (2007)], in which algorithms for merging ranked lists are widely used. Formally, the followings are defined:

TABLE 5.1: Score and Rank normalization methods

Method	Formula
Score Normalization	$w^{\tau_j}(x_i) = \frac{s^{\tau_j}(x_i) - min_j}{max_j - min_j}$
Z-Score Normalization	$w^{\tau_j}(x_i) = \frac{s^{\tau_j}(x_i) - \mu_{s^{\tau_j}}}{\sigma_{s^{\tau_j}}}$
Rank Normalization	$w^{\tau_j}(x_i) = 1 - \frac{\tau_j(x_i)-1}{\lvert\tau_j\rvert}$
Borda	$w^{\tau_j}(x_i) = \begin{cases} 1 - \frac{\tau_j(x_i)-1}{\lvert U\rvert} & \text{if } x_i \in \tau_j \\ \frac{1}{2} + \frac{\lvert\tau_j\rvert - 1}{2 \cdot \lvert U\rvert} & \text{otherwise} \end{cases}$

- U: the universe, that is the set containing all the distinct documents in the local lists;
- τ_j=$\{\ x_1 \geq x_2 \geq \ldots \geq x_n\ \}$: the j-th local list, $j = 1, \ldots, N$, defined as an ordered set S of documents, $S \subseteq U$, $\geq$ is the ranking criterion defined by the j-th local similarity function;
- $\tau_j(x_i)$: a function that returns the position of x_i in the list τ_j;
- $s^{\tau_j}(x_i)$: a function that returns the score of x_i in τ_j;
- $w^{\tau_j}(x_i)$: a function that returns the weight of x_i in τ_j.

  Two different strategies can be adopted to obtain $w^{\tau_j}(x_i)$, based on the score or the position of x_i in the list τ_j. Since local similarity functions may produce scores varying in different ranges, and the cardinality of lists can be different, a normalization process (of scores and positions) is necessary in order to produce weights that are comparable.

The aggregation of lists in a single one requires two steps: the first one produces the N normalized lists and the second one merges the N lists in a single one $\hat{\tau}$. In SENSE, both normalization strategies based on scores and positions. have been considered. Score normalization strategies compute $w^{\tau_j}(x_i)$ by using $s^{\tau_j}(x_i)$, while rank normalization strategies work on $\tau_j(x_i)$. Details are given in Table 5.1.

In the Score Normalization strategy, min_j is defined as $min_{x_k \in \tau_j} s^{\tau_j}(x_k)$; max_j is defined in an analogous way. While Score Normalization compares $w^{\tau_j}(x_i)$ to the minimum and the maximum scores in τ_j, Z-Score Normalization works on the average of the scores in τ_j, $\mu_{s^{\tau_j}}$, and their variance $\sigma_{s^{\tau_j}}$.

Rank normalization methods work by comparing the position of the document with respect to either the cardinality of the list to be normalized or the cardinality of the universe.

Given N normalized local lists τ_j, the goal of the rank aggregation method is to produce a new list $\hat{\tau}$, containing all documents in τ_j, ordered according to a *rank aggregation function* ψ that combines the normalized weights of local lists in a (hopefully) better ranking. Different strategies can be used to define ψ. Some of them are based on the concept of *rank hits* of a document x_i, that is the number of local lists which contain x_i. Let R be the set of all local lists, $R = \{\tau_1, \ldots, \tau_N\}$, $hits(x_i, R) = \mid \{\tau_j \in R : x_i \in \tau_j\} \mid$.

In SENSE the following rank aggregation methods have been adopted:
CombSUM - The score of document x_i in the global list is computed by summing all the normalized scores for x_i:

$$\psi(x_i) = \sum_{\tau_j \in R} w^{\tau_j}(x_i)$$

CombMNZ - It multiplies the CombSUM score by the rank hits, thus increasing the score of documents occurring in more than one local list:

$$\psi(x_i) = hits(x_i, R) \cdot \sum_{\tau_j \in R} w^{\tau_j}(x_i)$$

Weighted Combination - The score of document x_i in the global list is computed similarly to CombMNZ, except for the introduction of a boost factor α_j for each local list, in order to amplify (or reduce) the weight of x_i in each list:

$$\psi(x_i) = hits(x_i, R) \cdot \sum_{\tau_j \in R} \alpha_j \cdot w^{\tau_j}(x_i) \quad \sum \alpha_j = 1, \alpha_j \geq 0$$

where α_j underlines the importance of a local list in the global ranking, i.e. the importance of a level in SENSE. The motivation behind the choice is that CombSUM and CombMNZ operators have proved to perform better than others [Lee (1997)].

SENSE is a semantic N-levels IR system which manages documents indexed at multiple separate levels: keywords, senses and entities. The system is able to combine keyword search with semantic information provided by the two other indexing levels.
The distinctive feature of the system is that an IR framework is proposed to integrate, rather than simply replace, the lexical space with semantic

spaces. The prerequisite of such system is to have robust text processing, in order to semantically represent texts. Specifically WSD and NED techniques proved to add value to the semantic IR process. A WSD algorithm has been used to obtain the sense level, wether the NED methods object of this thesis have been used to identify entities which constitute the entity level.

5.2 Electronic Performance Support System

The NE management facilities implemented within the framework META, described in Chapter 3, have been exploited within an Electronic Performance Support System (EPSS), developed in the ambit of the project JUMP, *JUst-in-tiMe Performance support system for dynamic organizations*[5] [Iaquinta et al. (2007); Basile et al. (2007a,d)].

The JUMP project aims at bringing together the knowledge stored in different information systems in order to satisfy information and training needs in knowledge-intensive organisations. EPSSs provide help, advices, demonstrations, or any other informative support that a user needs to the accomplishment of job tasks in her day-to-day working environment. The JUMP framework is designed to offer multiple ways for the user to query the knowledge base resulting from integration of autonomous legacy systems. Semantic Web languages and technologies are used throughout the framework to represent, exchange and query the knowledge, while Natural Language Processing Techniques are implemented to understand natural language queries formulated by the user and provide consistent and satisfying results.

The EPSS developed within the JUMP project is capable of intelligent delivery of contextualized and personalized information to knowledge workers acting in their day-to-day working environment on non-routinary tasks. While generic queries can be easily fulfilled by means of stan-

[5]The JUMP project has been co-funded by Regione Puglia, Italy, through the research funding program named POR Puglia 2000-2006 - Mis. 3.13, Sostegno agli Investimenti in Ricerca Industriale, Sviluppo Precompetitivo e Trasferimento Tecnologico

dard information retrieval tools, such as general purpose search engines, the scenario is more diffcult if the search goal concerns grey information stored in various forms and spread in different company knowledge bases, managed by different applications, all running within the company intranet. It is the case of users knowledgeable w.r.t. the IT infrastructure and that already have the background knowledge necessary to achieve most of the task they are involved in, but not being expert of all the domains in which the task to be achieved spans. Tasks of this kind are neither generally codified in corporate procedures nor completely new to the worker. Above all, those tasks are by no means solvable, in terms of information retrieval, by a standard Internet search. Any brute-force approach like Google desktop search can solve the problem in this case and, even if it could, the result would never take into account the connections existing between the various sources. An EPSS aims at supporting information needs spanning through multiple knowledge bases, namely all the available information systems in the company, be them formalized or not, including binary documents such as video or audio streams. It acts as an agent gluing together the different sources by means of semantic connections, and provides the user with contextualized and personalized information tied to both the task being accomplished and to her characteristics. On the basis of an accurate and formalized description of user's features and of those of the software tool she is using, as well as the textual information describing the task being accomplished (for example, the text of an e-mail just received), the EPSS should select relevant items from the KBs, ranking them according to the user profile, and provide them in a list to the user who will eventually give a feedback about the relevance of the provided information. The JUMP system has been designed to achieve this goal by means of a centralized recommendation system that takes advantage of a shared ontology describing the various knowledge bases and advanced Natural Language Processing (NLP) techniques to handle natural language requests.

In the remaining of this Section it will be presented a general description of JUMP framework focusing on abstract layers of its architecture and the underlying shared ontology, a detailed description of the Content Analyzer Module encapsulated in the framework, which provides NLP tools.

5.2.1 JUMP: Ontology-centric Architecture

The system general architecture is depicted in Figure 5.2, where the central component (JUMP-EPSS) acts as a hub of many autonomous peripheral systems. The involved systems in the actual implementation are a Human Resources Managements system (HRMS), an Enterprise Resource Planning (ERP), a Document Management system (DMS) and a Learning Management system (LMS), but the design of the platform is such that new systems can be added as they become available.

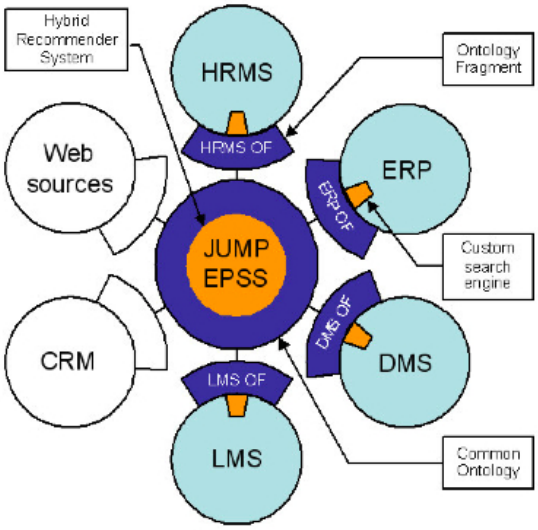

Figure 5.2: Sketch of the JUMP system architecture

The basic design idea of the JUMP project is to encourage the loosely coupling among framework components according to a Service Oriented Architecture perspective so that the framework has the ability to seamlessly add information sources or peripheral systems, each one based on different technologies, programming languages and knowledge representation metaphors. This has lead to adopting standard languages and protocols when designing the interfaces that each of the systems participating in JUMP has to implement in order to expose search services. The communication level between JUMP and the ancillary systems is designed

to exchange both metadata about relevant items stored in the subsystems and the items themselves. While the items considered here (which are the results JUMP can present to the user) are generic binary objects ranging from email addresses to audio/video streams, the metadata about them are expressed through Semantic Web technologies; to make this possible, some OWL[6] ontologies about the items have been created, in order to structure specific domain knowledge and instantiate resources to describe the stored items.

An ontology, following Grubers widely accepted definition [Gruber (1993)], is a shared formalization of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. In order to define an ontology it is necessary to choose a formalism, to use this formalism to encode the conceptualization that the applications are going to use, and to make this conceptualization shared, i.e. ensure that the ontology is used consistently by all the systems involved. To define the JUMP ontology, OWL has been adopted as representation language and Description Logics [Baader et al. (2003)] was consequently adopted as the underlying formalism. Separate ontology fragments have been handcrafted using the Protege editor[7] in order to represent the most important concepts inside each of the involved systems. Ontologies have been then designed bottom-up in order to reflect the semantics of the underlying databases and coded functional processes as much as possible, but not aiming at a total ontological replication of the knowledge bases. After developing the single Ontology Fragments (OF), they have been divided into system specific ontologies and upper ontologies; these upper ontologies are the part of the OFs that the JUMP system should use when formulating queries for the subsystems. The Shared Ontology (SO) is therefore the union of all the upper OFs plus all the relations and concepts that are specific to the JUMP system; since some concepts are repeated across systems, the creation of the SO is the point in which alignment techniques have to be used in order to simplify and generalize the query writing phase of the search. The concept in SO are annoted using lexical concept that are exploited by JIGSAW WSD

[6] http://www.w3.org/2004/OWL/
[7] http://protege.stanford.edu

algorithm [Basile et al. (2007b)]. The OFs are aligned manually using the concepts in the SO. The single OFs are populated automatically creating a mapping between each legacy system's DBMS and each fragment.

JUMP architecture has two abstract layers, as showed in figure 5.3, logically organized as a stack:

- User Interface Layer
- Application Domain Layer

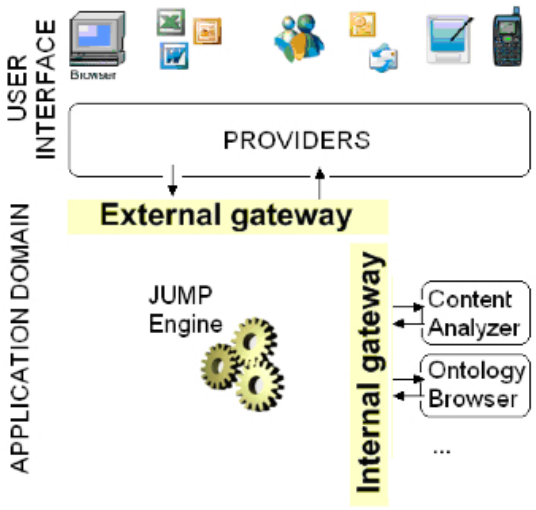

Figure 5.3: JUMP Engine and Framework Layers

The User Interface Layer has been designed to support both online and offline client. It supports several devices: The system handles both untrusted devices, such as web browser or Offce tools (Word/Excel), for which it has been predisposed a login procedure and trusted devices, such as mobile devices (e.g. smart phone) or email client for which the system is able to identify the user without any login procedure. The application domain layer has a software layer, namely external gateway, that communicates with all user devices through their specialized provider. The JUMP Engine elaborates all requests making use of specialized modules to resolve specific requests (e.g. plain text is sent to Content Analyzer Module, described in Section 5.2.2, in order to capture its semantics).

Communication between the JUMP Engine and all specialized modules is implemented through a software layer, namely internal gateway.

5.2.2 NLP Processes in the Content Analyzer Module

Since user's requests are formulated by using natural language, NLP techniques are adopted in order to covert the original requests into an internal representation processable by the JUMP system. The Content Analyzer Module is devoted to this task: it extracts relevant concepts from the text describing the request and build an internal data structure called Bag-Of-Concepts (BOC). The goal is to include semantics in the process and to overcome well-known problems in text processing, such as polysemy, due to the use of keywords. The BOC structure contains two type of concepts:

1. relevant *linguistic* concepts recognized in the text by a WSD process exploiting external linguistic knowledge-bases such as the WordNet lexical database;
2. relevant *domain* concepts extracted by a NER process. The NER process is guided by JUMP ontology.

As final output of the implemented NLP process each word in the original document is enriched with syntactic and semantic information.

The NER process for the aim of JUMP project cannot rely on general purpose gazetteers, due to specificity of categories and their instances for this particular project. For this reason it has been developed a simple algorithm to recognize entities using as gazetteers a domain ontology: each token in the original document is tagged with the ontology class value if it represents an instance of that class in the domain ontology. The idea of the algorithm follows. Given $C = \{C_1, C_2, \ldots, C_n\}$ the set of classes in the domain ontology, for each class C_k it is considered the set $P = \{p_1, p_2, \ldots, p_m\}$ of properties belonging to C_k . Given $T = \{t_1, t_2, \ldots, t_s\}$ the list of tokens obtained from document d, for each token t_j it is taken into account a window of h following tokens. The algorithm checks for each C_k if value of any combination of $t_j, \ldots, t_{j+h}$ matches with the value of any p_m, for all instances of C_k , and assigns to t_j the correspondent label. The search is done beginning from longer combinations of tokens and in the worst case it ends without any class annotation for the single

token t_j . Taking advantage of semantic information provided by ontology, relations between all entities found in the text can be simply obtained by exploiting the object properties defined within the ontology, without added computational cost.

Both the WSD procedure and the NER process are fundamental to obtain a concept-based text representation the Bag-Of-Concepts (BOC). In this model, a vector of concepts (WordNet synsets or named entities) corresponds to a document, instead of a vector of keywords. Therefore, each document (for example, an email text) representing the user request is converted in a BOC structure in order to be processed by the JUMP Engine. A more formal description of the BOC text representation follows. Assuming of having a document d_n (corresponding to a user's request n) processed by the Content Analyzer Module. The document is converted into the following BOC structure
$d_n =< (t_{n1}, wn_1), (t_{n2}, w_{n2}), ..., (t_{n|V|}, w_{n|V|}) >$ where:

- t_{nk} is the $k-th$ token (synset or named entity) recognized in document dn by NLP procedures;
- V is the set of distinct tokens recognized in d_n;
- w_{nk} is the weight representing the informative power of token t_{nk} in document d_n.

In other words, a text is represented by a vector of pairs (token, weight), where tokens are recognized from keywords in the text by NLP procedures which assign each token with a numerical score representing the discriminatory power of that token in the text. Weights can be computed in different ways for synsets and NE. For example, let us consider a user who is going to prepare a technical report for a project. She has technical skills but no idea about how to compile such kind of document. She queries the JUMP System, using the following expression

q : have to prepare a technical report for the JUMP project.

The JUMP Engine passes the user's request q to the Content Analyzer module that processes the document in order to both disambiguate the task to be accomplished, by using the WSD procedure, and to recognize entities potentially useful for the task. The final output of this stage is q represented according to the BOC model:

$q =< (01704982, 0.75), (06775158, 0.85), (JUMP, 0.99), (00746508, 0.80) >$ where both synset identifiers of the concepts recognized in the text and NE are used in the BOC structure. For example, the verb *prepare* has been disambiguated as the synset reported below:

```
01704982 (verb.creation) prepare -- to prepare verbally,
either for written or spoken delivery; "prepare a report"
```

which identifies the task to be accomplished. This structure is used to query the Shared Ontology. As a result, the JUMP Engine provides all istances of the concepts technical report and project and relations intercourring among these instances and istances of different classes of the ontology. In the example, the system returns the list of partners and documents related to the JUMP project, and the list of technical reports already written for other projects.

The presented JUMP system shows how NLP methods can be effective in real applications. JUMP constitutes the implementation of a framework for knowledge sharing within knowledge-intensive organizations by personalized information retrieval. The user current task and background knowledge are used to fulfill informative request. NLP and shared domain ontology are exploited to semantic interpretation of user request, which is expected to be a simple text query (not different from a normal query that could be issued against a standard query engine such as Google or Yahoo), in order to query legacy knowledge bases.

Chapter 6

Conclusions and Future Work

This thesis work proposed two novel methods for Named Entity Disambiguation. Both proposed methods, named respectively WibNED and SRaNED, are knowledge based methods but while WibNED is merely builded on word overlapping, SRaNED builds an entity graph, taking into account more sophisticated features from the Knowledge Base, and use such graph to calculate Semantic Relatedness between entities within the text. Experiments showed that a promising direction is that of using Semantic Relatedness measures for the process of NED.

A consideration must be done on the decision of using Wikipedia as *entity inventory*: in terms of lexical coverage, especially with reference to specific domains, the choice impose some limitations. It is the case of specific technical areas, such as Medicine, Biology, specific fields of engineering, etc., which can have little or no coverage within the Wikipedia source. However both methodologies are still applicable when the *entity inventory* is a different KB, with a few requirements, first of all that the KB has coverage for entities of interest. For WibNED algorithm it is also necessary to capture a short textual description for each entity: as long as it is possible to extract such information from the KB, the method is easily adaptable. For the SRaNED algorithm the only additional requirement is that it is possible to calculate relatedness scores between entities within the KB: e.g. if Wikipedia is changed with a domain ontology it could be possible to define measure of relatedness within the ontology and, after having relatedness scores between entities, applying the same

proposed NED step. Theoretically we might expect this to be true but obviously, experiments are needed to prove the efficacy of such approach and future work could follow this direction.

Future directions

A first direction for further investigation is focusing on Specific Domain, by adapting proposed methods on different Knowledge Bases. All NLP techniques taken into account within this dissertation have been regarded as functional for the purpose of automatically *understanding* the text. A more ambitious mission for NLP techniques is that of reading the intentions behind the text: the research field of *Sentiment Analysis* deals with the computational treatment of sentiments, subjectivity and opinions within a text. Opinion mining or extraction is a research topic at the crossroads of information retrieval and computational linguistics concerned with enabling automatic systems to determine human opinion from text written in natural language. Recently, opinion mining has gained much attention due to the explosion of user-generated content on the Web. Automatically tracking attitudes and feelings in on-line blogs and forums, and obtaining first hand reactions to the news on-line, is a desirable instrument to support statistical analyses by companies, the government, and even individuals. An interaction between Opinion Mining and Named Entities is easily expectable: automatically extracting people's opinion about companies, specific products, singers, politicians could be a valuable facility. An innovative exploitation could be, e.g., extracting opinions about specific persons within Social Networks.

Appendix A
HMM Example

Formally a HMM can be defined as a tuple: $< \mathbf{S}, \mathbf{W}, \pi, \mathbf{A}, \mathbf{B} >$, where

- $\mathbf{S} = \{\mathbf{s_1}, ..., \mathbf{s_n}\}$ is the set of states in the model;
- $\mathbf{W} = \{\mathbf{w_1}, ..., \mathbf{w_n}\}$ is the set of observations or output symbols;
- π are a-priori likelihoods, that is the initial state distribution:
  $\pi = \{\pi_\mathbf{i}\}$, $\mathbf{i} \epsilon \mathbf{S}$;
- A = $\{\mathbf{a_{ij}}\}, \mathbf{i}, \mathbf{j} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}\}$, is a matrix describing the state transition probability distribution:
  $\mathbf{a_{ij}} = \mathbf{P}(\mathbf{X_{t+1}} = \mathbf{s_j} | \mathbf{X_t} = \mathbf{s_i})$;
- $\mathbf{B} = \{\mathbf{B_{ij_k}}\}, \mathbf{i}, \mathbf{j} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}\}, \mathbf{k} \epsilon \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{t}\}$, is a matrix describing the observation symbol probability distribution:
  $\mathbf{b_{ij_k}} = \mathbf{P}(\mathbf{O_t} = \mathbf{w_k} | \mathbf{X_t} = \mathbf{s_i}, \mathbf{X_{t+1}} = \mathbf{s_j})$;

Let us consider the problem of guessing the weather in a city based on a weekly report on how many passports were lost.
We have two possible states in the model, $S = \{Sunny(S), Rainy(R)\}$. Given that $\pi(X)$ indicates the start probability, $A(X, Y)$ the transition probability and $B(X, Y)$ the emission probability, let us consider the city of London. We have a probability $\pi(S) = 0.55$ for the weather being sunny and $\pi(R) = 0.45$ for the weather being rainy. When the weather is sunny it is more likely that the day after is sunny again $A(S, S) = 0.75$,

instead of being rainy $A(S,R) = 0.25$. When the weather is rainy it is more likely that the day after is rainy again $A(R,R) = 0.7$, instead of being sunny $A(R,S) = 0.3$.
The possible output is constituted by $W = \{L, \#\}$, where L indicates the fact some passports get lost and # that no passports get lost during the day . The emission probability for L and # in this example, with respect to the weather state (S or R) is set as:

- $B(S,L) = 0.7$, $B(S,\#) = 0.3$
- $B(R,L) = 0.2$, $B(R,\#) = 0.8$

If it is assumed to have a report of $\#\#L$, i.e. it is observed that for the first two days no passport get lost and some passports get lost the third day.

What it is needed to predict with the HMM model is the most likely sequence of weather stases that produce the output $\#\#L$.
We can find the most probable sequence of hidden states by listing all possible sequences of hidden states and finding the probability of the observed sequence for each of the combinations. We need to calculate the probability of all possible combinations of weather states for a sequence of three days, based on the observations of passports lost reports:

$$(S,S,S->S) = \pi(S)*A(S,S)*B(S,\#)*A(S,S)*B(S,\#)*A(S,S)*$$
$$B(S,L) = 0.5500*0.7500*0.3000*0.7500*0.3000*0.7500*0.7000 = 0.0146$$

$$(S,S,S->R) = \pi(S)*A(S,S)*B(S,\#)*A(S,S)*B(S,\#)*A(S,R)*$$
$$B(S,L) = 0.5500*0.7500*0.3000*0.7500*0.3000*0.2500*0.7000 = 0.0049$$

$$(S,S,R->S) = \pi(S)*A(S,S)*B(S,\#)*A(S,R)*B(S,\#)*A(R,S)*$$
$$B(R,L) = 0.5500*0.7500*0.3000*0.2500*0.3000*0.3000*0.2000 = 0.0006$$

$$(S,S,R->R) = \pi(S)*A(S,S)*B(S,\#)*A(S,R)*B(S,\#)*A(R,R)*$$
$$B(R,L) = 0.5500*0.7500*0.3000*0.2500*0.3000*0.7000*0.2000 = 0.0013$$

$$(S,R,S->S) = \pi(S)*A(S,R)*B(S,\#)*A(R,S)*B(R,\#)*A(S,S)*$$
$$B(S,L) = 0.4500*0.2500*0.3000*0.3000*0.8000*0.7500*0.7000 = 0.0043$$

$(S, R, S- > R) = \pi(S) * A(S, R) * B(S, \#) * A(R, S) * B(R, \#) * A(S, R) *$
$B(S, L) = 0.4500 * 0.2500 * 0.3000 * 0.3000 * 0.8000 * 0.2500 * 0.7000 = 0.0014$

$(S, R, R- > S) = \pi(S) * A(S, R) * B(S, \#) * A(R, R) * B(R, \#) * A(R, S) *$
$B(R, L) = 0.4500 * 0.2500 * 0.3000 * 0.7000 * 0.8000 * 0.3000 * 0.2000 = 0.0011$

$(S, R, R- > R) = \pi(S) * A(S, R) * B(S, \#) * A(R, R) * B(R, \#) * A(R, R) *$
$B(R, L) = 0.4500 * 0.2500 * 0.3000 * 0.7000 * 0.8000 * 0.7000 * 0.2000 = 0.0026$

$(R, S, S- > S) = \pi(R) * A(R, S) * B(R, \#) * A(S, S) * B(S, \#) * A(S, S) *$
$B(S, L) = 0.5500 * 0.3000 * 0.8000 * 0.7500 * 0.3000 * 0.7500 * 0.7000 = 0.0156$

$(R, S, S- > R) = \pi(R) * A(R, S) * B(R, \#) * A(S, S) * B(S, \#) * A(S, R) *$
$B(S, L) = 0.5500 * 0.3000 * 0.8000 * 0.7500 * 0.3000 * 0.2500 * 0.7000 = 0.0052$

$(R, S, R- > S) = \pi(R) * A(R, S) * B(R, \#) * A(S, R) * B(S, \#) * A(R, S) *$
$B(R, L) = 0.5500 * 0.3000 * 0.8000 * 0.2500 * 0.3000 * 0.3000 * 0.2000 = 0.0006$

$(R, S, R- > R) = \pi(R) * A(R, S) * B(R, \#) * A(S, R) * B(S, \#) * A(R, R) *$
$B(R, L) = 0.5500 * 0.3000 * 0.8000 * 0.2500 * 0.3000 * 0.7000 * 0.2000 = 0.0014$

(R, R, S -> $\mathbf{S)} = \pi(R) * A(R, R) * B(R, \#) * A(R, S) * B(R, \#) * A(S, S) *$
$B(S, L) = 0.4500 * 0.7000 * 0.8000 * 0.3000 * 0.8000 * 0.7500 * 0.7000 =$
0.0318

$(R, R, S- > R) = \pi(R) * A(R, R) * B(R, \#) * A(R, S) * B(R, \#) * A(S, R) *$
$B(S, L) = 0.4500 * 0.7000 * 0.8000 * 0.3000 * 0.8000 * 0.2500 * 0.7000 = 0.0106$

$(R, R, R- > S) = \pi(R) * A(R, R) * B(R, \#) * A(R, R) * B(R, \#) * A(R, S) *$
$B(R, L) = 0.4500 * 0.7000 * 0.8000 * 0.7000 * 0.8000 * 0.3000 * 0.2000 = 0.0085$

$(R, R, R- > R) = \pi(R) * A(R, R) * B(R, \#) * A(R, R) * B(R, \#) * A(R, R) *$
$B(R, L) = 0.4500 * 0.7000 * 0.8000 * 0.7000 * 0.8000 * 0.7000 * 0.2000 = 0.0198$

The combination which maximize the probability is $(R, R, S- > S) = 0.0318$ and it is concluded that the most likely series of weather reports is Rainy, Rainy, Sunny.

Appendix B
Viterbi Algorithm

Taking a particular HMM, we usually want to determine from an observation sequence the most likely sequence of underlying hidden states that might have generated it. We can find the most probable sequence of hidden states by listing all possible sequences of hidden states and finding the probability of the observed sequence for each of the combinations (as showed in Appendix A). This approach is viable, but to find the most probable sequence by exhaustively calculating each combination is computationally expensive. We can use the time invariance of the probabilities to reduce the complexity of the calculation.

The Viterbi algorithm can be seen as a way of finding the shortest route through a graph. The Markov process can be represented as a state diagram, like one shown in figure 6.1, where nodes (circles) represent states, arrows represent transitions, and over the course of time the process traces some path from state to state through the state diagram.

A more redundant description of the same process, called trellis, is shown in figure 6.2: each node corresponds to a distinct state at a given time, and each arrow represents a transition to some new state at the next instant of time. The trellis begins and ends at the known states c_0 and c_n. Its most important property is that to every possible state sequence C there corresponds a unique path through the trellis, and vice versa.

The function $forward\_viterbi$ takes the following arguments: obs is the sequence of observations; $states$ is the set of hidden states; $start_p$is the start probability; $trans\_p$ are the transition probabilities; and $emit\_p$

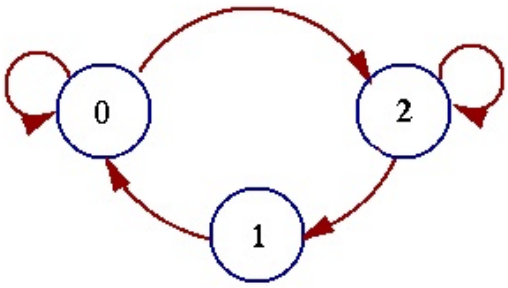

FIGURE 6.1: Markov process representation: State Diagram

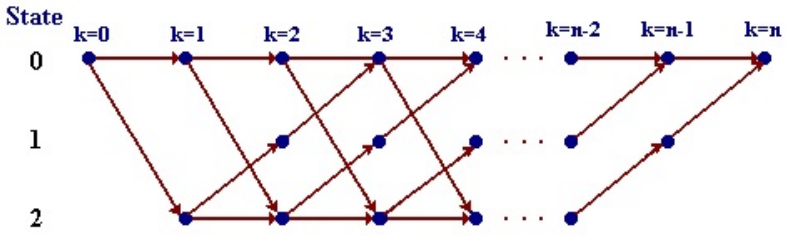

FIGURE 6.2: Markov process representation: Trellis

Algorithm 2 Viterbi Algorithm

procedure $forward\_viterbi(obs, states, start\_p, trans\_p, emit\_p)$
  $S = \{\}$
  for all $state \in states$ **do**
    $T[state] = (start\_p[state], [state], start\_p[state])$
  end for
  for all $output \in obs$ **do**
    $U = \{\}$
    for all $next\_state \in states$ **do**
      $total = 0$
      $argmax = None$
      $valmax = 0$
      for all $source\_state \in states$ **do**
        $(prob, v\_path, v\_prob) = T[source\_state]$
        $p = emit\_p[source\_state][output] * trans\_p[source\_state][next\_state]$
        $prob* = p$
        $v\_prob* = p$
        $total+ = prob$
        if $v\_prob > valmax$ **then**
          $argmax = v\_path + [next\_state]$
          $valmax = v\_prob$
        end if
      end for
      U[next_state] = (total, argmax, valmax)
    end for
    T = U
  end for
  $total = 0$
  $argmax = None$
  $valmax = 0$
  for all $state \in states$ **do**
    $(prob, v\_path, v\_prob) = T[state]$
    $total+ = prob$
    if $v\_prob > valmax$ **then**
      $argmax = v\_path$
      $valmax = v\_prob$
    end if
  end for
  return $(total, argmax, valmax)$
end procedure

are the emission probabilities.

The algorithm works on the mappings T and U. Each is a mapping from a *state* to a triple $(prob, v\_path, v\_prob)$, where *prob* is the total probability of all paths from the start to the current state (constrained by the observations), v_path is the *Viterbi path* up to the current state, and v_prob is the probability of the *Viterbi path* up to the current state. The mapping T holds this information for a given point t in time, and the main loop constructs U, which holds similar information for time $t + 1$. Because of the Markov assumption, which states that the probability of a state occurring given a previous state sequence depends only on the previous n states, in particular, with a first order Markov assumption, the probability of X occurring after a sequence depends only on the previous state, i.e. , information about any point in time prior to t is not needed.

The algorithm begins by initializing T to the start probabilities: the total probability for a state is just the start probability of that state; and the *Viterbi path* to a start state is the singleton path consisting only of that state; the probability of the *Viterbi path* is the same as the start probability.

The main loop considers the observations from *obs* in sequence. Its loop invariant is that T contains the correct information up to but excluding the point in time of the current observation. The algorithm then computes the triple $(prob, v\_path, v\_prob)$ for each possible next state. The total probability of a given next state, *total*, is obtained by adding up the probabilities of all paths reaching that state. More precisely, the algorithm iterates over all possible source states. For each source state, T holds the total probability of all paths to that state. This probability is then multiplied by the emission probability of the current observation and the transition probability from the source state to the next state. The resulting probability *prob* is then added to total. The probability of the *Viterbi path* is computed in a similar fashion, but instead of adding across all paths one performs a discrete maximization. Initially the maximum value valmax is zero. For each source state, the probability of the *Viterbi path* to that state is known. This too is multiplied with the emission and transition probabilities and replaces *valmax* if it is greater than its current value. The *Viterbi path* itself is computed as the corresponding *argmax* of that maximization, by extending the Viterbi path that leads to

the current state with the next state. The triple $(prob, v\_path, v\_prob)$ computed in this fashion is stored in U and once U has been computed for all possible next states, it replaces T, thus ensuring that the loop invariant holds at the end of the iteration.

Let us consider the same settings as the example on HMM, in Appendix A, and we want to use the Viterbi algorithm to guess the most likely sequence of weather states in a city, based on a weekly report on how many passports were lost. The settings are:

- States: $S = \{Sunny(S), Rainy(R)\}$
- Start probability: $\pi(S) = 0.55$, $\pi(R) = 0.45$
- Transition probability:
  - $A(S, S) = 0.75$, $A(S, R) = 0.25$
  - $A(R, R) = 0.7$, $A(R, S) = 0.3$
- Output: $W = \{L, \#\}$
  (L some passports lost, $\#$ no passports lost)
- Emission probability:
  - $B(S, L) = 0.7$, $B(S, \#) = 0.3$
  - $B(R, L) = 0.2$, $B(R, \#) = 0.8$

If it is assumed to have a report of $\# \# L$, i.e. it is observed that for the first two days no passport get lost and some passports get lost the third day. Differently as previous example, we do not need to perform an exhaustive computation of probabilities, but we consider three subsequent instant of time t_1, t_2, t_3.
At istant t_0 the algorithm considers as start score the total probability for each state. The score at t_0 for the states Sunny and Rainy are:

- $score(S, 0) = 0.55$ (which coincides with $\pi(S) = 0.55$)
- $score(R, 0) = 0.45$ (which coincides with $\pi(R) = 0.45$)

At instant t_1 the score for the states Sunny and Rainy are:

- $score(S, 1) = 0.1237$
  $MAX(candidate\_score(S, S, 1), candidate\_score(R, S, 1))$
  - $candidate\_score(S, S, 1) = score(S, 0) * A(S, S) * B(S, \#) = 0.5500 * 0.7500 * 0.3000 = 0.1237$

  - $candidate\_score(R, S, 1) = score(R, 0) * A(R, S) * B(R, \#) = 0.4500 * 0.3000 * 0.8000 = 0.1080$

- $score(R, 1) = 0.2520$
  $MAX(candidate\_score(S, R, 1), candidate\_score(R, R, 1))$

  - $candidate\_score(S, R, 1) = score(S, 0) * A(S, R) * B(S, \#) = 0.5500 * 0.2500 * 0.3000 = 0.0413$
  - $candidate\_score(R, R, 1) = score(R, 0) * A(R, R) * B(R, \#) = 0.4500 * 0.7000 * 0.8000 = 0.2520$

At instant t_2 the score for the states Sunny and Rainy are:

- $score(S, 2) = 0.0605$
  $MAX(candidate\_score(S, S, 2), candidate\_score(R, S, 2))$

  - $candidate\_score(S, S, 2) = score(S, 1) * A(S, S) * B(S, \#) = 0.1237 * 0.7500 * 0.3000 = 0.0278$
  - $candidate\_score(R, S, 2) = score(R, 1) * A(R, S) * B(R, \#) = 0.2520 * 0.3000 * 0.8000 = 0.0605$

- $score(R, 2) = 0.1411$
  $MAX(candidate\_score(S, R, 2), candidate\_score(R, R, 2))$

  - $candidate\_score(S, R, 2) = score(S, 1) * A(S, R) * B(S, \#) = 0.1237 * 0.2500 * 0.3000 = 0.0093$
  - $candidate\_score(R, R, 2) = score(R, 1) * A(R, R) * B(R, \#) = 0.2520 * 0.7000 * 0.8000 = 0.1411$

At instant t_3 the score for the states Sunny and Rainy are:

- $score(S, 3) = 0.0318$
  $MAX(candidate\_score(S, S, 3), candidate\_score(R, S, 3))$

  - $candidate\_score(S, S, 3) = score(S, 2) * A(S, S) * B(S, L) = 0.0605 * 0.7500 * 0.7000 = 0.0318$
  - $candidate\_score(R, S, 3) = score(R, 2) * A(R, S) * B(R, L) = 0.1411 * 0.3000 * 0.2000 = 0.0085$

- $score(R, 3) = 0.0198$
  $MAX(candidate\_score(S, R, 3), candidate\_score(R, R, 3))$

- $candidate\_score(S, R, 3) = score(S, 2) * A(S, R) * B(S, L) = 0.0605 * 0.2500 * 0.7000 = 0.0106$
- $candidate\_score(R, R, 3) = score(R, 2) * A(R, R) * B(R, L) = 0.1411 * 0.7000 * 0.2000 = 0.0198$

Table 6.1: Viterbi computation example

	$t = 0$	$t = 1$	$t = 2$	$t = 3$
Sunny	**0.55**	0.1237	0.0605	**0.0318**
Rainy	0.45	**0.2520**	**0.1411**	0.0198

The most likely series of weather reports is Rainy, Rainy, Sunny.

The Viterbi algorithm provides a computationally efficient way of analysing observations of HMMs to recapture the most likely underlying state sequence. It exploits recursion to reduce comuputational load, and uses the context of the entire sequence to make judgements, thereby allowing good analysis of noise.

Appendix C
Decision Tree Example

This Appendix reports two example of decision tree, one proposed by Sekine (1998) and the other by Pailouras et al. (2000), both using C4.5 [Quinlan (1993)] for the learning process.

Table 6.2: Japanese Sentence Example

Token	POS	Char-type	Special Dict.	NE answer
ISURAERU	PN-loc	Kata	loc	org-OP-CN
KEISATSU	N	Kanji	org-S	org-CN-CL
NI	postpos	Hira		
YORU	V	Hira		
TO	postpos	Hira		
,	comma	Comma		
ERUSAREMU	PN-loc	Kata	loc	loc-OP-CN

Table 6.2 shows an example of Japanese sentence used by Sekine (1998). The table contains the sentence along with three types of information (part-of-speech, character type and special dictionary information) and given information of opening and closing of named entities. Character type information indicate the writing system. In the japanese language there are three different writing system: Katakana (characterized by short, straight strokes and angular corners), hiragana (more rounded

than katakana) and kanji (of chinese origin). Special Dictionaries information states if the word belongs to a special list of entities (created based on the Web or based on human knowledge).

TABLE 6.3: Example of Decision Tree for NER

ISURAERU (first token)
if current token is a location → yes
if next token is a loc-suffix → no
if next token is a person-suffix → no
if next token is a org-suffix → yes
if previous token is a location → no
THEN none = 0.67, org-OP-CN = 0.33

KEISATSU (second token)
if current token is a location → no
if current token is a organization → no
if current token is a time → no
if current token is a loc-suffix → no
if next token is a time-suffix → no
if current token is a time-suffix → no
if next token is a date-suffix → no
if current token is a date-suffix → no
if current token is a date → no
if next token is a location → no
if current token is a org-suffix → yes
if previous token is a location → yes
THEN none= 0.14 org-CN-CL = 0.86

The decision tree gives an output for each token. It is one of the 4 possible combinations of opening, continuation and closing a named entity and having no named entity. It is built using training data. The model learns about the opening and closing of named entities based on

the three kinds of information (part-of-speech, character type and special dictionary information) of the previous, current and following tokens.

Table 6.3 shows two example paths in the decision tree. For the purpose of the example, the first and second token of the example sentence in table 6.2 have been used. Each line corresponds to a question asked by the tree nodes along the path. The last line shows the probabilities of named entity information which have more than 0.0 probability. The probability of none for the first token (Isuraeru = Israel) is higher than that for the opening of organization (0.67 to 0.33) but in the second token (Keisatsu = Police) the probability of closing organization is much higher than none (0.86 to 0.14).

Another example of using decision trees for the task of NER is proposed by Pailouras et al. (2000) (it is a partial NER, because they only recognize person and organization). They used C4.5 [Quinlan (1993)] to learn the decision tree. C4.5 requires training data to be represented in a feature-vector format.
For the task of NER, a training instance is a named-entity (NE) phrase, consisting of one or more words, plus some external information, i.e., words in the close neighbourhood of the NE phrase. Two features are used for each word: its gazetteer tag, if it has one, and its part of speech. The feature vector consists of 14 words: 10 words for the NE phrase plus the two adjacent words on each side of the phrase. Therefore, each vector consists of 28 features, 14 part-of-speech and 14 gazetteer tags. When the NE phrase is shorter than 10 words, the remaining features are assigned a special value (a question mark), treated as a label for missing information by the algorithm. Words that are not be in it, in order for it to be a named entity.

Figure 6.3 presents a set of rules, which correspond to a decision tree of 58 nodes that was generated in their experiments. There are three possible classes: person, organisation and non-NE.

Representative classifier: (abbreviations: POS=part-of-speech tag, Gtag=gazetteer tag)

IF Gtag(-1) **IN** {`title, org_key+title`} **THEN** *person*
ELSEIF Gtag(-1) **IN** {`NOTAG, currency_unit, date, location, org_key+organisation, organisation, person`} **THEN:**
  IF Gtag(1) **=** `NOTAG` **THEN:**
    IF POS(1) **IN** {`NNP, VBG`} **THEN**:
      IF POS(+2) **IN** {`RP, VB, WP`} **THEN** *person*
      ELSEIF POS(+2) **IN** {`CD, MD, NN, NNS, RB, TO`}
        AND POS(-1) **IN** {`CC, NN, PERIOD, SYM, VB, VBD, VBZ`}
        THEN *person*
      ELSE *organisation*
    ELSE *non-NE*
  ELSEIF Gtag(1) **IN** {`cdg, govern_key, location, location+title, org_base, org_key, title`} **THEN:**
    IF POS(1) **IN** {`NNP, VBG`} **AND** POS(-1) **IN** {`DT, JJ`} **THEN** *organisation*
    ELSE *non-NE*
  ELSEIF Gtag(1) **IN** {`location+organisation, org_key+organisation, organisation+person`} **THEN:**
    IF Gtag(+1) **IN** {`cdg, organisation, person`} **THEN** *non-NE*
    ELSEIF Gtag(+1) = `NOTAG` **THEN:**
      IF POS(+1) **IN** {`CC, COMMA, DT, IN, JJ, JJR, NN, NNS, POS, SYM, TO, VB, VBD, VBZ, WP`} **THEN:**
        IF POS(4) **IN** {`CC, IN, JJ, NN, NNS, VBD, VBZ`}
          AND POS(2) **IN** {`COMMA, IN, NN, NNS, POS`}
          AND POS(-2) **IN** {`CC, CD, NN, NNS, PERIOD, PRP, VB, VBZ, WP, JJ, NNP, SYM, TO, VBD, VBN`}
          THEN *non-NE*
        ELSE *organisation*
      ELSEIF POS(+1) **IN** {`CD, MD, NNP, NNPS, PERIOD, RB, VBG, VBN, VBP`} **THEN:**
        IF POS(-1) **IN** {`DT, JJ`} **THEN** *organisation*
        ELSEIF POS(-1) **IN** {`CC, COMMA, NNP, PERIOD, PPS, SYM, TO, VBN`} **AND** POS(+2) **IN** {`NN, VBZ`} **THEN** *organisation*
        ELSE *non-NE*
      ELSE *organisation*
    ELSE *organisation*
  ELSEIF Gtag(1) **=** `person` **THEN:**
    IF POS(+1) **IN** {`CD, NNP, VBP`} **THEN** *non-NE*
    ELSE *person*
  ELSE *non-NE*
ELSE *non-NE*

FIGURE 6.3: An example of Decision Tree for NER

Bibliography

Agirre, E. and Lopez de Lacalle Lekuona, O. (2004). Publicly available topic signatures for all wordnet nominal senses. In *Proceedings of the Fourth International Conference on Languages Resources and Evaluations, LREC 2004. Lisbon, Portugal*.

Agirre, E., Martínez, D., López de Lacalle, O., and Soroa, A. (2006). Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 585–593, Sydney, Australia. Association for Computational Linguistics.

Alfonseca, E. and Manandhar, S. (2002). An unsupervised method for general named entity recognition and automated concept discovery. In *Poceedings of the First International Conference on General WordNet*, Mysore, India.

Aswani, N., Bontcheva, K., and Cunningham, H. (2006). Mining information for instance unification. In Cruz et al. (2006), pages 329–342. ISBN 3-540-49029-9.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. ISBN 0-521-78176-0.

Bagga, A. and Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85, Morristown, NJ, USA. Association for Computational Linguistics.

Banerjee, S. and Pedersen, T. (2002). An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 136–145, London, UK. Springer-Verlag. ISBN 3-540-43219-1.

Basile, P., Caputo, A., de Gemmis, M., Gentile, A. L., Lops, P., and Semeraro, G. (2008a). Improving Ranked Keyword Search with SENSE: SEmantic N-levels Search Engine. *Communications of SIWN (formerly: System and Information Sciences Notes) SIWN: The Systemics and Informatics World Network*, pages 39–45.

Basile, P., Caputo, A., Gentile, A. L., de Gemmis, M., Lops, P., and Semeraro, G. (2008b). Enhancing semantic search using n-levels document representation. In Bloehdorn, S., Grobelnik, M., Mika, P., and Tran, D. T., editors, *SemSearch*, volume 334 of *CEUR Workshop Proceedings*, pages 29–43. CEUR-WS.org.

Basile, P., Caputo, A., Gentile, A. L., de Gemmis, M., Lops, P., and Semeraro, G. (2008c). Improving Retrieval Experience Exploiting Semantic Representation of Documents. In *Semantic Web Applications and Perspectives (SWAP2008), Rome, Italy, December 15-17, 2008*, volume 426. CEUR Workshop Proceedings.

Basile, P., de Gemmis, M., Gentile, A. L., Iaquinta, L., Lops, P., and Semeraro, G. (2007a). An Electronic Performance Support System Based on a Hybrid Content-Collaborative Recommender System. *Neural Network World: International Journal on Neural and Mass-Parallel Computing and Information Systems*, 17(6):529–541.

Basile, P., de Gemmis, M., Gentile, A. L., Iaquinta, L., Lops, P., and Semeraro, G. (2008d). META - MultilanguagE Text Analyzer. In *Proceedings of the Language and Speech Technnology Conference - LangTech 2008, Rome, Italy, February 28-29*, pages 137–140.

Basile, P., de Gemmis, M., Gentile, A. L., Lops, P., and Semeraro, G. (2007b). The JIGSAW Algorithm for Word Sense Disambiguation and Semantic Indexing of Documents. In Basili, R. and Pazienza, M. T.,

editors, *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence, Rome, Italy, September 10-13, 2007, Proceedings*, volume 4733 of *Lecture Notes in Computer Science*, pages 314–325. Springer. ISBN 978-3-540-74781-9.

Basile, P., de Gemmis, M., Gentile, A. L., Lops, P., and Semeraro, G. (2007c). Uniba: Jigsaw algorithm for word sense disambiguation. In *Proceedings of the 4th ACL 2007 International Worshop on Semantic Evaluations (SemEval-2007)*, pages 398–401. Association for Computational Linguistics (ACL).

Basile, P., de Gemmis, M., Iaquinta, L., Gentile, A. L., and Lops, P. (2007d). The JUMP project: domain ontologies and linguistic knowledge @ work. In Semeraro, G., Sciascio, E. D., Morbidoni, C., and Stoermer, H., editors, *SWAP*, volume 314 of *CEUR Workshop Proceedings*, pages 61–70. CEUR-WS.org.

Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Bick, E. (2004). A Named Entity Recognizer for Danish. In *Proceedings of the Fourth International Conference on Languages Resources and Evaluations, LREC 2004. Lisbon, Portugal.*

Bikel, D., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: a high-performance learning name-finder. In *Proc. of the 5th conference on Applied Natural Language Processing (ANLP 97).*

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

Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. *SIGMOD Rec.*, 30(2):175–186.

Borthwick, A. E. (1999). *A maximum entropy approach to named entity recognition.* PhD thesis, New York University, New York, NY, USA. ISBN 0-599-47232-4.

Brin, S. (1998). Extracting patterns and relations from the world wide web. In Atzeni, P., Mendelzon, A. O., and Mecca, G., editors, *WebDB*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183. Springer. ISBN 3-540-65890-4.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*.

Bunescu, R. C. and Pasca, M. (2006). Using Encyclopedic Knowledge for Named entity Disambiguation. In *EACL*. The Association for Computer Linguistics. ISBN 1-932432-59-0.

Califf, M. E. and Mooney, R. J. (2003). Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.*, 4:177–210.

Chen, Y. and Martin, J. (2007). Towards robust unsupervised personal name disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 190–198, Prague, Czech Republic. Association for Computational Linguistics.

Chieu, H. L. and Ng, H. T. (2003). Named Entity Recognition with a Maximum Entropy Approach. In *In Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003*, pages 160–163.

Chinchor, N. (1998). Overview of MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

Cohen, W. W. and Sarawagi, S. (2004). Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, New York, NY, USA. ACM. ISBN 1-58113-888-1.

Collins, M. (2002). Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 489–496, Morristown, NJ, USA. Association for Computational Linguistics.

Collins, M. and Singer, Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors (2006). *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-49029-9.

Cucerzan, S. (2007). Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic. Association for Computational Linguistics.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and Weischedel, R. (2004). The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *Proceedings of the Fourth International Conference on Languages Resources and Evaluations, LREC 2004. Lisbon, Portugal*, pages 837–840.

Etzioni, O., Cafarella, M. J., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134.

Farah, M. and Vanderpooten, D. (2007). An outranking approach for rank aggregation in information retrieval. In Kraaij, W., de Vries, A. P., Clarke, C. L. A., Fuhr, N., and Kando, N., editors, *SIGIR*, pages 591–598. ACM. ISBN 978-1-59593-597-7.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Society*, 64(328):1183–1210.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: the concept revisited. In *ACM Transactions on Information Systems*, volume 20 (1), pages 116–131.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

García, N. F., del Toro, J. M. B., Sánchez, L., and Bernardi, A. (2007). IdentityRank: Named Entity Disambiguation in the Context of the NEWS Project. In Franconi, E., Kifer, M., and May, W., editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 640–654. Springer. ISBN 978-3-540-72666-1.

Gentile, A. L., Basile, P., Iaquinta, L., and Semeraro, G. (2008). Lexical and semantic resources for nlp: From words to meanings. In Lovrek, I., Howlett, R. J., and Jain, L. C., editors, *KES (3)*, volume 5179 of *Lecture Notes in Computer Science*, pages 277–284. Springer. ISBN 978-3-540-85566-8.

Gentile, A. L., Basile, P., and Semeraro, G. (2009). WibNED: Wikipedia Based Named Entity Disambiguation. In Agosti, M., Esposito, F., and Thanos, C., editors, *IRCDL*, pages 51–59. DELOS: an Association for Digital Libraries. ISBN 978-88-903541-7-5.

Ghani, R., Jones, R., and Mladenic, D. (2001). Mining the web to create minority language corpora. In Paques et al. (2001), pages 279–286. ISBN 1-58113-436-3.

Giles, J. (2005). Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901.

Grishman, R. and Sundheim, B. (1996). Message Understanding Conference- 6: A Brief History. In *COLING*, pages 466–471.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. In *Knowledge Engineering, 5(2)*, pages 199–220. Academic Press.

Han, H., Giles, C. L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In Chen, H., Wactlar, H. D., chih Chen, C., Lim, E.-P., and Christel, M. G., editors, *JCDL*, pages 296–305. ACM. ISBN 1-58113-832-6.

Han, H., Zha, H., and Giles, C. L. (2003). A model-based k-means algorithm for name disambiguation. In *Proceedings of the 2nd International Semantic Web Conference (ISWC-03) Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*.

Han, X. and Zhao, J. (2009). Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 215–224, New York, NY, USA. ACM. ISBN 978-1-60558-512-3.

Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

Hassell, J., Aleman-Meza, B., and Arpinar, I. B. (2006). Ontology-driven automatic entity disambiguation in unstructured text. In Cruz et al. (2006), pages 44–57. ISBN 3-540-49029-9.

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H., and Wilks, Y. (1998). University of sheffield: Description of the lasie-ii system as used for muc-7. In *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*. Morgan.

Iaquinta, L., Gentile, A. L., Lops, P., de Gemmis, M., and Semeraro, G. (2007). A Hybrid Content-Collaborative Recommender System Integrated into an Electronic Performance Support System. In Koenig, A., Köppen, M., Abraham, A., and Kasabov, N., editors, *Proceedings of the Seventh International Conference on Hybrid Intelligent Systems HIS-2007*, pages 47–52. IEEE Computer Society Press, Los Alamitos, California. ISBN 0-7695-2946-1.

Iria, J., Xia, L., and Zhang, Z. (2007). Wit: Web people search disambiguation using random walks. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 480–483, Prague, Czech Republic. ACL.

Jaynes, E. T. (1957). Information theory and statistical mechanics export. *Physical Review Online Archive (Prola)*, 106(4):620–630.

Kalashnikov, D. V. and Mehrotra, S. (2005). A probabilistic model for entity disambiguation using relationships. In *SIAM International Conference on Data Mining (SDM)*.

Kazama, J. and Torisawa, K. (2007). Exploiting wikipedia as external knowledge for named entity recognition. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Kilgarriff, A. and Grefenstette, G. (2003). Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–348.

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., and Collier, N. (2004). Introduction to the bio-entity recognition task at jnlpba. In *JNLPBA '04: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75, Morristown, NJ, USA. Association for Computational Linguistics.

Kripke, S. A. (1972). *Naming and Necessity*. Harvard University Press. ISBN 0-674-59846-6.

Krupka, G. R. and Hausman, K. (1998). Isoquest inc.: Description of the netowltm extractor system as used for muc-7. In Chinchor, N. A., editor, *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*, Fairfax, Virginia. Morgan.

Kudo, T. and Matsumoto, Y. (2003). Fast Methods for Kernel-Based Text Analysis. In Hinrichs, E. and Roth, D., editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 24–31.

Kwok, C., Etzioni, O., and Weld, D. (2001). Scaling question answering to the web. *ACM Trans. Inf. Syst.*, 19(3):242–262.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1.

Leacock, C. and Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. In Fellbaum, C., editor, *WordNet: An Electronic Lexical Database*, pages 265–283. MIT Press.

Lee, J.-H. (1997). Analyses of Multiple Evidence Combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–276. ACM.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of ACM SIGDOC Conference*, pages 24–26.

Lin, D. (2000). Word sense disambiguation with a similarity-smoothed case library. *Computers and the Humanities*, 34(1-2):147–152.

Lovász (1996). Random walks on graphs: A survey. *Combinatorics, Paul Erdös is Eighty*, 2:353–398.

Magnini, B. and Cavaglià, G. (2000). Integrating subject field codes into wordnet. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000. Athens, Greece*, pages 1413–1418.

Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2002). Is it the right answer? exploiting web redundancy for answer validation. In *ACL*, pages 425–432.

Malin, B. (2005). Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*.

Mann, G. S. and Yarowsky, D. (2003). Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 33–40, Morristown, NJ, USA. Association for Computational Linguistics.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 188–191. Edmonton, Canada.

Mihalcea, R. (2005). Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *HLT/EMNLP*. The Association for Computational Linguistics.

Mikheev, A., Grover, C., and Moens, M. (1998). Description of the LTG system used for MUC-7. In Chinchor, N. A., editor, *Proceedings of Seventh Message Understanding Conference (MUC–7)*, Fairfax, Virginia.

Mikheev, A., Moens, M., and Grover, C. (1999). Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 1–8, Bergen, Norway.

Miller, G. A. (1995). WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41.

Minkov, E., Cohen, W. W., and Ng, A. Y. (2006). Contextual search and name disambiguation in email using graphs. In Efthimiadis, E. N., Dumais, S. T., Hawking, D., and Järvelin, K., editors, *SIGIR*, pages 27–34. ACM. ISBN 1-59593-369-7.

Minkov, E., Wang, R., and Cohen, W. W. (2005). Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT-EMNLP*.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification.

Nadeau, D., Turney, P., and Matwin, S. (2006). Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. *Advances in Artificial Intelligence*, pages 266–277.

Navigli, R. (2006). Meaningful clustering of senses helps boost word sense disambiguation performance. In *Annual Meeting of the Association for Computational Linguistics joint with the 21st International Conference on Computational Linguistics (COLING-ACL 2006)*, pages 105–112.

Navigli, R. and Lapata, M. (2007). Graph connectivity measures for unsupervised word sense disambiguation. In Veloso, M. M., editor, *IJCAI*, pages 1683–1688.

Newcombe, H., Kennedy, J., Axford, S., and James, A. (1959). Automatic linkage of vital records. *Science*, 130:954Ð959.

Nguyen, H. T. and Cao, T. H. (2008). Named entity disambiguation on an ontology enriched by Wikipedia. In *RIVF*, pages 247–254. IEEE.

Nie, Z., Zhang, Y., Wen, J., and Ma, W. (2005). Object-level ranking: bringing order to web objects. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 567–574, New York, NY, USA. ACM. ISBN 1-59593-046-9.

Nuray-Turan, R., Kalashnikov, D. V., and Mehrotra, S. (2007). Self-tuning in graph-based reference disambiguation. In Ramamohanarao, K., Krishna, P. R., Mohania, M. K., and Nantajeewarawat, E., editors, *DASFAA*, volume 4443 of *Lecture Notes in Computer Science*, pages 325–336. Springer. ISBN 978-3-540-71702-7.

Okanohara, D., Miyao, Y., Tsuruoka, Y., and ichi Tsujii, J. (2006). Improving the scalability of semi-markov conditional random fields for named entity recognition. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *ACL*. The Association for Computer Linguistics.

Pailouras, G., Karkaletsis, V., and Spyropoulo, C. D. (2000). Learning decision trees for named-entity recognition and classification. In Ciravegna, F., Basili, R., and Gaizauskas, R., editors, *Proceedings of the ECAI workshop on Machine Learning for Information Extraction, held at the 14th European Conference on Artificial Intelligence (ECAI2000)*.

Paques, H., Liu, L., and Grossman, D., editors (2001). *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001*. ACM. ISBN 1-58113-436-3.

Patrick, J., Whitelaw, C., and Munro, R. (2002). Slinerc: the sydney language-independent named entity recogniser and classifier. In *COLING-02: proceedings of the 6th conference on Natural language learning*, pages 1–4, Morristown, NJ, USA. Association for Computational Linguistics.

Pedersen, T., Purandare, A., and Kulkarni, A. (2005). Name discrimination by clustering similar contexts. In Gelbukh, A. F., editor, *CICLing*, volume 3406 of *Lecture Notes in Computer Science*, pages 226–237. Springer. ISBN 3-540-24523-5.

Peng, Y., He, D., and Mao, M. (2006). Geographic named entity disam-

biguation with automatic profile generation. In *Web Intelligence*, pages 522–525. IEEE Computer Society. ISBN 0-7695-2747-7.

Pietra, V. D., Pietra, V. D., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

Ponzetto, S. P. and Strube, M. (2006). Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In Moore, R. C., Bilmes, J. A., Chu-Carroll, J., and Sanderson, M., editors, *HLT-NAACL*. ACL.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296. ISBN 1-55860-124-4.

Radev, D., Qi, H., Zheng, Z., Blair-Goldensohn, S., Zhang, Z., Fan, W., and Prager, J. (2001). Mining the web for answers to natural language questions. In Paques et al. (2001), pages 143–150. ISBN 1-58113-436-3.

Ramshaw, L. and Marcus, M. (1995). Text chunking using transformation-based learning. In Yarovsky, D. and Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Rau, L. F. (1991). Extracting company names from text. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, volume i, pages 29–32.

Rich, E. (1972). *Automata, Computability, and Complexity: Theory and Applications*. Prentice Hall. ISBN 9780132288064.

Riloff, E. and Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 474–479,

Menlo Park, CA, USA. American Association for Artificial Intelligence. ISBN 0-262-51106-1.

Rosso, P., y Gómez, M. M., Buscaldi, D., Pancardo-Rodríguez, A., and Pineda, L. V. (2005). Two web-based approaches for noun sense disambiguation. In *CICLing*, pages 267–279.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Santos, D., Seco, N., Cardoso, N., and Vilela, R. (2006). HAREM: An Advanced NER Evaluation Contest for Portuguese. In Calzolari, N., Choukri, K., Gangemi, A., Maegaard, B., Mariani, J., Odjik, J., and Tapias, D., editors, *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006*, pages 1986–1991, Genoa, Italy. ELRA.

Sarawagi, S. and Cohen, W. W. (2004). Semi-Markov Conditional Random Fields for Information Extraction. In *NIPS*.

Satoshi Sekine, C. N. (2004). Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the Fourth International Conference on Languages Resources and Evaluations, LREC 2004. Lisbon, Portugal*, Marrakech, Morocco. European Language Resources Association (ELRA).

Searle, J. R. (1958). Proper Names. *Mind*, 67(266):166–173.

Sekine, S. (1998). Nyu: Description of the japanese ne system used for met-2. In *Proceedings of the Message Understanding Conference*.

Sekine, S. and Isahara, H. (2000). IREX: IR and IE evaluation-based project in Japanese. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000. Athens, Greece*.

Semeraro, G., de Gemmis, M., Lops, P., and Basile, P. (2007). Combining Learning and Word Sense Disambiguation for Intelligent User Profiling. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence IJCAI-07*, pages 2856–2861. M. Kaufmann, San Francisco, California. ISBN 978-I-57735-298-3.

Settles, B. (2004). Biomedical named entity recognition using conditional random fields and rich feature sets. In *JNLPBA '04: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107, Morristown, NJ, USA. Association for Computational Linguistics.

Shinyama, Y. and Sekine, S. (2004). Named entity discovery using comparable news articles. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 848, Morristown, NJ, USA. Association for Computational Linguistics.

Sinha, R. and Mihalcea, R. (2007). Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC*, pages 363–369. IEEE Computer Society.

Srinivasan, H., Chen, J., and Srihari, R. (2009). Cross document person name disambiguation using entity profiles. In *Proceedings of IJCAI 2009 Workshop on Information Integration on the Web*.

Strapparava, C. and Valitutti, A. (2004). Wordnet-affect: an affective extension ofwordnet. In *Proceedings of the Fourth International Conference on Languages Resources and Evaluations, LREC 2004. Lisbon, Portugal*, pages 1083–1086.

Strube, M. and Ponzetto, S. P. (2006). WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *AAAI*, pages 1419–1424. AAAI Press.

Sugiyama, K. and Okumura, M. (2007). Personal name disambiguation in web search results based on a semi-supervised clustering approach. In Goh, D. H.-L., Cao, T. H., Sølvberg, I., and Rasmussen, E. M., editors, *ICADL*, volume 4822 of *Lecture Notes in Computer Science*, pages 250–256. Springer. ISBN 978-3-540-77093-0.

Terra, E. and Clarke, C. (2003). Frequency estimates for statistical word similarity measures. In *HLT-NAACL*.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Toral, A. and Munoz, R. (2006). A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. In *Workshop on New Text, 11th Conference of the European Chapter of the Association for Computational Linguistics. Trento (Italy). April 2006.*

Turdakov, D. and Velikhov, P. (2008). Semantic relatedness metric for wikipedia concepts based on link analysis and its application to word sense disambiguation. In Kuznetsov, S. D., Pleshachkov, P., Novikov, B., and Shaporenkov, D., editors, *SYRCoDIS*, volume 355 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Turney, P. (2001). Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In Raedt, L. D. and Flach, P., editors, *ECML*, volume 2167 of *Lecture Notes in Computer Science*, pages 491–502. Springer. ISBN 3-540-42536-5.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 0-387-94559-8.

Vercoustre, A.-M., Thom, J. A., and Pehcevski, J. (2008). Entity ranking in wikipedia. In Wainwright, R. L. and Haddad, H., editors, *SAC*, pages 1101–1106. ACM. ISBN 978-1-59593-753-7.

Wacholder, N., Ravin, Y., and Choi, M. (1997). Disambiguation of proper names in text. In *Proceedings of the 17th Annual ACM-SIGIR Conference*, pages 202–208.

Wu, S. Y. S. B. P. (1998). Description of the kent ridge digital labs system used for muc-7. In *Proceedings of the Message Understanding Conference*.

Yangarber, R., Lin, W., and Grishman, R. (2002). Unsupervised learning of generalized names. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Yusuke, M. and Jun'ichi, T. (2002). Maximum entropy estimation for feature forests. In *Proceedings of the second international conference on Human Language Technology Research*, pages 292–297, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zesch, T., Gurevych, I., and Mühlhäuser, M. (2007). Analyzing and Accessing Wikipedia as a Lexical Semantic Resource. In *Biannual Conference of the Society for Computational Linguistics and Language Technology*.

Zesch, T., Müller, C., and Gurevych, I. (2008). Using wiktionary for computing semantic relatedness. In Fox, D. and Gomes, C. P., editors, *AAAI*, pages 861–866. AAAI Press. ISBN 978-1-57735-368-3.

Zhou, G. and Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480, Morristown, NJ, USA. Association for Computational Linguistics.

Ziff, P. (1960). *Semantic Analysis*. Cornell University Press, Ithaca, NY, USA. ISBN 0-801-49051-0.