EXPLOITING KNOWLEDGE IN NLP

BY

LEV RATINOV

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Abstract

In recent decades, the society depends more and more on computers for a large number of tasks. The first steps in NLP applications involve identification of topics, entities, concepts, and relations in text. Traditionally, statistical models have been successfully deployed for the aforementioned problems. However, the major trend so far has been: scaling up by dumbing down- that is, applying sophisticated statistical algorithms operating on very simple or low-level features of the text. This trend is also exemplified, by expressions such as *"we present a knowledge-lean approach"*, which have been traditionally viewed as a positive statement, one that will help papers get into top conferences. This thesis suggests that it is essential to use knowledge in NLP, proposes several ways of doing it, and provides case studies on several fundamental NLP problems.

It is clear that humans use a lot of knowledge when understanding text. Let us consider the following text *"Carnahan campaigned with Al Gore whenever the vice president was in Missouri."* and ask two questions: (1) who is the vice president? (2) is this sentence about politics or sports? A knowledge-lean NLP approach will have a great difficulty answering the first question, and will require a lot of training data to answer the second one. On the other hand, people can answer both questions effortlessly.

We are not the first to suggest that NLP requires knowledge. One of the first such large-scale efforts, CYC[1] , has started in 1984, and by 1995 has consumed a person-century of effort collecting 10^5 concepts and 10^6 commonsense axioms, including *"You can usually see peoples noses, but not their hearts."*. Unfortunately, such an effort has several problems. (a) The set of facts we can deduce is significantly larger than 10^6 . For example, in the above example *heart* can be replaced by any internal organ or tissue, as well as by a bank account, thoughts etc., leading to thousands of axioms. (b) The axioms often do not hold. For example, if the person is standing with their back to you, can cannot see their nose. And during an open heart surgery, you can see someone's heart. (c) Matching the concepts to natural-language expressions is challenging. For example, *Al Gore* can be referred to as *Democrat, environmentalist, vice president, Nobel prize laureate* among other things. The idea of *buying a used car* can be also expressed as *purchasing a pre-owned automobile*. The

[1]While CYC had a very broad set of applications in mind, NLP, Machine Translation, and Information Retrieval were some of the key motivations, as stated in [78]

lexical variability in text makes using knowledge challenging. Instead of focusing on *obtaining* a large set of logic axioms, we are focusing on *using* knowledge-rich features in NLP solutions.

We have used three sources of knowledge: a large corpus of unlabeled text, encyclopedic knowledge derived from Wikipedia and first-order-logic-like constraints within a machine learning framework. Namely, we have developed a Named Entity Recognition system which uses word representations induced from unlabeled text and gazetteers extracted from Wikipedia to achieve new state of the art performance. We have investigated the implications of augmenting text representation with a set of Wikipedia concepts. The concepts can either be directly mentioned in the documents, or not explicitly mentioned but closely related. We have shown that such document representation allows more efficient search and categorization than the traditional lexical representations. Our next step is using the knowledge injected from Wikipedia for co-reference resolution. While the majority of the knowledge in this thesis is encyclopedic, we have also investigated how knowledge about the structure of the problem in the form of constraints can allow leveraging unlabeled data in semi-supervised settings.

This thesis shows how to use knowledge to improve state-of-the-art for four fundamental problems in NLP: text categorization, information extraction, concept disambiguation and coreference resolution, four tasks which have been considered the bedrock of NLP since its inception.

To my friends and foes.

Acknowledgments

I want to thank all of those who have shaped me as a researcher and a human being. Chronologically, perhaps I should start by thanking Anton Loginov, who called me a nerd and refused to let me into his soccer team when I was in the third grade. Perhaps he helped me more than anybody else to unveil my true nature and get into the world of books and thoughts. With his help, I realized I have to be Lev Ratinov, not a wannabe Lev Yashin. Watching my friends grow, I feel that giving up on pretending to be someone else since an early age gave me almost an unfair advantage over my peers.

Also, having moved from Kyrgyzstran to Israel and then to USA, I have to thank all the people whom I culturally shocked, and who have culturally shocked me. This made me re-evaluate my ways, hopefully to the better. Reuven Danino, thanks for calling me a "stinky Russian" and giving me a black eye, while suffering a broken nose, this lesson is still cherished 18 years later. I know that what you wanted to say was much deeper than that, and you just wanted to make me a more acceptable person in a society. Anyway, I'm taking a better care of my hygiene these days. Thank you, the American pickup artists who have helped me unlock the intricacies of the American dating and helped me to better understand what it means to be a man. With my improved hygiene, I can really see the results of your coaching... I guess everything is finally coming together now, everything was meant to be.

I owe most of the success in my life (professional and personal) to people who saw the goodness in me at a young age, often when I and everybody else failed to see it. Talent hits a target no one else can hit; Genius hits a target no one else can see. My mentors were genius in seeing the goodness and talents hidden in me and went out of their way to pass me their knowledge and experience. Thank you, my neighbor Holocaust survivor Genya, my wise chess coach Yossef Kagan, my youth mentor Eli Partish, my English teacher and mentor Victor Khalepski, my mom's friend Amnon Yitzhaki who taught me history and Derech Eretz, my early Math mentor Arkady Leiderman, my later Research mentors - Ehud Gudes, Eyal Shimony and Mark Last.

With time, I became increasingly proud of my old-time academic brothers from BGU: Yohai Twitto, Assaf Klein, Guy Shani, Wisam Daka, Fadi Biadsy, Gabriel Zaccak, Yoav Goldberg, to name a few. It makes me feel very lucky to have that kind of background. I appreciate people of great knowledge and rootedness who took their time to say a kind word to me, especially Michael Elhadad.

I want to thank the warmasters from the Taekwondo scene who taught me discipline, dedication and what being resolved really means. Thank you Eyal Fima, Alex Man, Eran Wass, Menahem Neumann, Shai Shitrit, Shani Hecht, Yan Fraiman, and of course, the Fuchs brothers, who cannot agree whether their last name should be Fuks or Fux. Special thanks to Tim Ghormley the coach of the Stanford Taekwondo program, I'm still keeping the belt you took off your waist and gave me.

Thank you to my army peers, commanders and students, you taught me humility and comradeship while generously allowing me pursue my academic and Taekwondo goals. Big thanks to people who helped me step beyond Taekwondo and become a collaborative rather than a competitive human being, most and foremost Igal Yachtengerg, Hila Zvilling, Dani Nevet.

I want to thank all the administrative office people who helped my studies tick in Israel and in the USA, especially Eric Bailey, Holly Ann Bagwell, Mary-Beth Kelley, Mazal Gagulashvili, Aviva Levi, Zahava Kubani. Your help behind the scenes did not go unnoticed.

Thanks to people who helped me when I was most vulnerable: Alexander Binshtok, who helped me to recover from my first knee injury and also taught me so much about compassion and about being a giver. Thank you Yael Karmi for saving me in the middle of the night, in the middle of nowhere when I injured my other knee. Yael, you also helped me find peace with myself and with my parents, you exposed me to the art of non-violent communication, you changed my life. Special thanks to Max Stessel, who has been a source of positive inspiration in my final year of Phd.

I found life in Urbana Champaign hard and dull and the people who lit it up, mean the world to me. Bonnie Nozari, ever since you've graduated and left, Urbana-Champaign has looked like nothing to me but a deserted rusty car wreck on the side of the road to pass by and to momentarily look at with sadness. Kate Insolia, without the creative outlets with you, I'd surely go crazy. Dan Goldwasser, your tranquility and depth have enriched my life in so many ways. Some of my brightest memories from the Phd period are our conversations and trips. Antoine Kalfi and Celina Aguilar - I carry a piece of you in my heart too.

My labmates have made my stay in Urbana more tolerable and very fruitful. Special thanks to Ming-Wei Chang, the co-author of my first Phd paper. Thank you Nick Rizollo for being so available with technical help. Quang Do, Kevin Small, Dan Goldwasser, Alla Rozovskaya, Yuancheng Tu, Vivek Srikumar, Rajhans Samdani, , Mark Summons, Kai-Wei Chang.

Special thanks go to my official and unofficial mentors throughout my internship stints: Moises Goldszmidt, Kannan Achan, Rakesh Gupta, Baris Yuksel, Raoul-Sam Daruwala, Ryan McDonald. Each of you has taught me a special skill that is unique to you. Big thanks to my co-authors who taught me research discipline and prowess, especially Hiroya Takamua, Joseph Turian, Ming-Wei Chang and Doug Downey. Likewise, special thanks go to my thesis committee: Rada Michalcea, ChengXiang Zhai, Jiawei Han.

I'm forever indebted to people who had me in their mercy and who showed amazing understanding and kindness - my advisor Dan Roth and my TA boss Cinda Heeren. You were more than just understanding and kind but also tolerant, patient and forgiving. You are a venerable and rare example of brilliant people who also excel in being a loving parent, spouse and friend. I *admire* and *revere* both of you. You've been an inspiration for me in so many ways and you give me faith in the human kind.

Finally, and most importantly, I want to thank those who gave me life and raised me. My dad, Victor has taught me how to be inquisitive, perceptive, kind and ubiquitously loving. My mom, Elena has taught me how to accept the Truth even when it's not pleasant and how to be determined, brave, uncompromising and hard working. Thank you my grandma Sonya for connecting me to my roots. Without knowing who I am and where I'm coming from, I'd surely be running around like a lost puppy looking for superior Bagels and Matzo Ball soup. Most importantly, I thank both of my parents for teaching me how to deal with and accept what may appear unfair in life. You've always taught me by example - by stoically and lovingly overcoming incredible adversities and hardships.

Each word in this thesis is a collaborative effort by all who have left an imprint in my soul, from Anton Loginov to Victor Ratinov. Big thank you, everybody!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BOW	Bag of words.
CODL	Constraint-drivel learning.
GLOW	Global and local Wikification.
IDF	Inverse document frequency.
SVM	Support vector machine.
TF	Term frequency.
TFIDF	Term frequency–inverse document frequency.
NB	Naive Bayes.
NER	Named entity recognition.

Chapter 1

Introduction

In recent decades, the society depends more and more on computers for a large number of tasks. We are using social networks to learn about activities and interests of our friends, we use search engines to find documents. We rely on online encyclopedias to learn about concepts and events. As our interaction with the electronic environment becomes more involved and our knowledge moves to electronic format, the necessity of processing written information intelligently also increases. For example, modern email clients can remind us when we want to add attachment and forgot to do it. Lawsuits over patent infringements involve analyzing hundreds of thousands of electronic documents. Running data mining algorithms on medical records across thousands of patients is necessary to understand the patterns of (mis)diagnosis, symptoms and diseases. Automatic redirection of customer complaints to the correct customer service representative is a common practice for effective business operation.

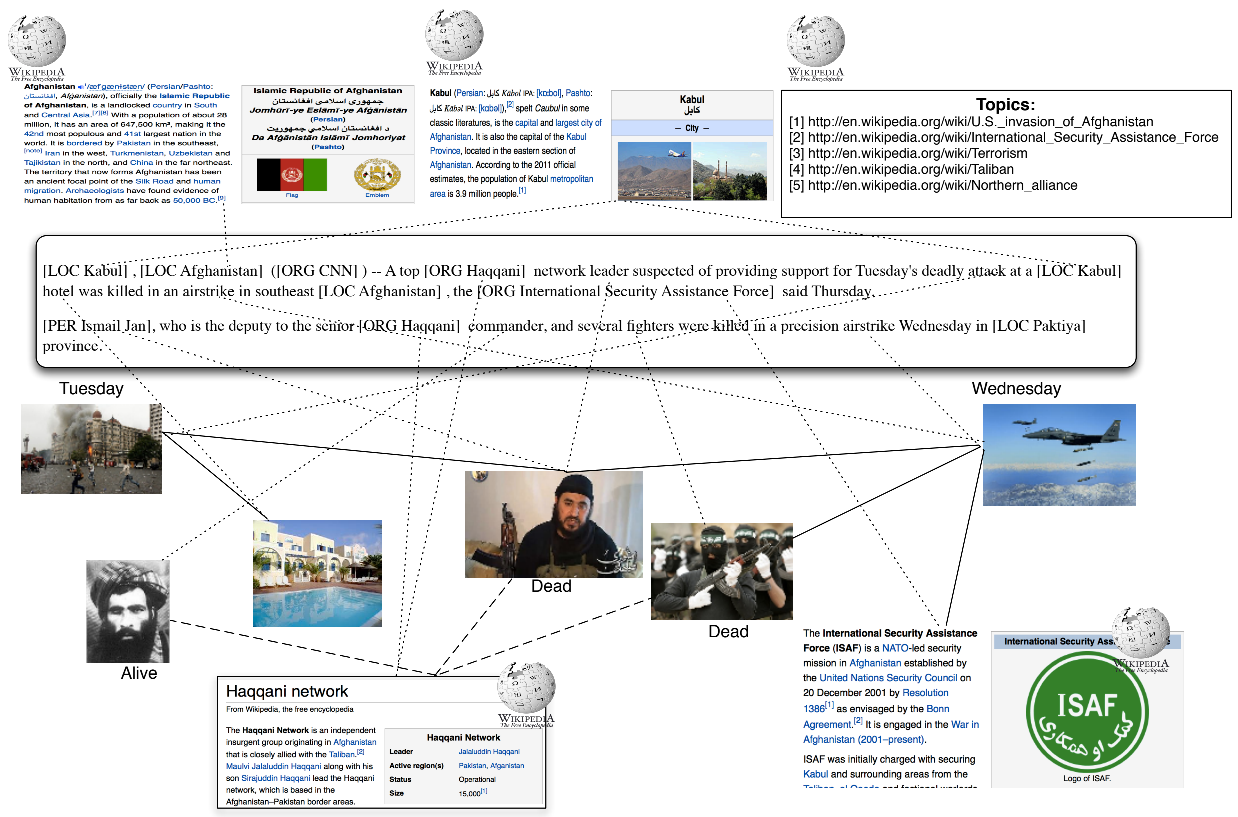
1.1 Challenges of Language Understanding

What does it take to understand language? Let us consider the the first two sentences from a CNN article[1] given in Figure 1.1. We may be surprised by the amount of information a human can extract from these two sentences alone. First, we can answer the question about the general topics of the article: the U.S. invasion of Afghanistan and terrorism. Then, we can identify the named entities: the locations (Kabul, Afghanistan, Paktiya province), the organizations (CNN, Haqqani network, International Security Assistance Force), and people (Ismail Jan). We note that all but one entity have the corresponding Wikipedia pages where the reader can find additional information. We also note that the majority of the readers would know that Kabul is the capital of Afghanistan and would have some prior knowledge about the activities and the agenda of the Haqqani network as well as the International Security Assistance Force. The reader of the article can also understand that "Ismail Jan", a "senior Haqqani commander", and "several fighters" are members of the Haqqani network. We also learn about two events that occurred: "a deadly attack at a

[1]http://www.cnn.com/2011/WORLD/asiapcf/06/30/afghanistan.kabul.attack/index.html?hpt=wo_c1

Kabul hotel" and a "precision airstrike". The later attack has killed "Ismail Jan", who is also mentioned as "top Haqqani, network leader", but not the anonymous "senior Haqqani commander" to whom Ismail Jan acts as a deputy. Therefore, one can expect more deadly attacks on Kabul hotels and more retaliation airstrikes.

In the natural language processing community, the task identifying the topics of the article is what we refer to as *text categorization*, identifying the entities, such as "Ismail Jan" is called *named entity recognition*, mapping the entities to Wikipedia pages is called *concept resolution*. Understanding that "Ismail Jan" and "top Haqqani, network leader" refer to the same entity, but distinct from "senior Haqqani commander" is called *co-reference resolution*. In this thesis, we focus on improving the state of the art for these problems.



Afghanistan /æfˈɡænɨstæn/ (Persian/Pashto: افغانستان, *Afġānistān*), officially the **Islamic Republic of Afghanistan**, is a landlocked country in South and Central Asia.[7][8] With a population of about 28 million, it has an area of 647,500 km², making it the 42nd most populous and 41st largest nation in the world. It is bordered by Pakistan in the southeast,[note] Iran in the west, Turkmenistan, Uzbekistan and Tajikistan in the north, and China in the far northeast. The territory that now forms Afghanistan has been an ancient focal point of the Silk Road and human migration. Archaeologists have found evidence of human habitation from as far back as 50,000 BC.[9]

Islamic Republic of Afghanistan
جمهوری اسلامی افغانستان
Jomhūrī-ye Eslāmī-ye Afġānistān
(**Persian**)
د افغانستان اسلامي جمهوريت
Da Afġānistān Islāmī Jomhoriyat
(**Pashto**)

Kabul (Persian: کابل *Kābol* IPA: [kɒːbol], Pashto: کابل *Kābəl* IPA: [kɑbəl]),[2] spelt *Caubul* in some classic literatures, is the capital and largest city of Afghanistan. It is also the capital of the Kabul Province, located in the eastern section of Afghanistan. According to the 2011 official estimates, the population of Kabul metropolitan area is 3.9 million people.[1]

Topics:
[1] http://en.wikipedia.org/wiki/U.S._invasion_of_Afghanistan
[2] http://en.wikipedia.org/wiki/International_Security_Assistance_Force
[3] http://en.wikipedia.org/wiki/Terrorism
[4] http://en.wikipedia.org/wiki/Taliban
[5] http://en.wikipedia.org/wiki/Northern_alliance

[LOC Kabul] , [LOC Afghanistan] ([ORG CNN]) -- A top [ORG Haqqani] network leader suspected of providing support for Tuesday's deadly attack at a [LOC Kabul] hotel was killed in an airstrike in southeast [LOC Afghanistan] , the [ORG International Security Assistance Force] said Thursday.

[PER Ismail Jan], who is the deputy to the senior [ORG Haqqani] commander, and several fighters were killed in a precision airstrike Wednesday in [LOC Paktiya] province.

Haqqani network
From Wikipedia, the free encyclopedia

The **Haqqani Network** is an independent insurgent group originating in Afghanistan that is closely allied with the Taliban.[2] Maulvi Jalaluddin Haqqani along with his son Sirajuddin Haqqani lead the Haqqani network, which is based in the Afghanistan–Pakistan border areas.

Haqqani Network	
Leader	Jalaluddin Haqqani
Active region(s)	Pakistan, Afganistan
Status	Operational
Size	15,000[1]

The **International Security Assistance Force** (**ISAF**) is a NATO-led security mission in Afghanistan established by the United Nations Security Council on 20 December 2001 by Resolution 1386[1] as envisaged by the Bonn Agreement.[2] It is engaged in the War in Afghanistan (2001–present).

ISAF was initially charged with securing Kabul and surrounding areas from the

Figure 1.1: Sample text illustrating the challenges of natural language understanding.

1.2 The Role of Knowledge in NLP

So far, statistical models have been successfully deployed for the aforementioned problems. However, the major trend so far has been: scaling up by dumbing down- that is, applying sophisticated statistical algorithms operating on very simple or low-level features of the text. This trend is also exemplified, by expressions such as *"we present a knowledge-lean approach"*, which have been traditionally viewed as a positive statement, one that will help papers to get into top conferences. However, our analysis of the text in Figure 1.1 shows that humans use a lot of knowledge when understanding text. While knowledge-lean approaches are attractive, they are ultimately limited, and as the amount of on-line text and information resources increases, it creates an opportunity to utilize them for machine-learning purposes.

We believe that humans solve NLP tasks jointly, while using a lot of world knowledge. For instance, in our running example, a human can infer that Haqqani network is a terrorist organization. And since the airstrike has killed Ismail Jan, he is dead. For a computer, making these "simple" inferences is exceptionally challenging since the current state of the art in natural language processing does not have adequate tools for modeling common-sense knowledge, narrative and discourse structure, semantics and pragmatics.

We are not the first to suggest that NLP requires knowledge. One of the first such large-scale efforts, CYC[2] , has started in 1984, and by 1995 has consumed a person-century of effort collecting 10^5 concepts and 10^6 commonsense axioms, such as *"You can usually see peoples noses, but not their hearts."*. Unfortunately, such an effort has several problems. (a) The set of facts we can deduce is significantly larger than 10^6 . For example, in the above example *heart* can be replaced by any internal organ or tissue, as well as by a bank account, thoughts etc. Transitivity can also be a factor. For example, if a medicine can relieve headache, and Tylenol is a medicine, then Tylenol may (or may not, and here non-probabilistic inference falls apart) relieve headache. Inference with this amount of knowledge is beyond the current state of the art computationally (b) The axioms often do not hold. For example, if the person is standing with their back to you, can cannot see their nose. And during an open heart surgery, you can see someone's heart. (c) Matching the concepts to natural-language expressions is challenging. For example, *Al Gore* can be referred to as *Democrat, environmentalist, vice president, Nobel prize laureate* among other things. The idea of *buying a used car* can be also expressed as *purchasing a pre-owned automobile*. The lexical variability in text makes using knowledge challenging. This is the main reason why CYC, as well as many other attempts to manually curate knowledge, such as OpenMind [131], WordNet [43], FrameNet [3] have had a very limited impact in NLP.

With little progress made in applying manually curated resources in practical NLP applications, much of the research community has turned its attention to automatically learning a large set of facts from unlabeled text [41; 4; 161; 115; 127; 17]. The hope is that automatic fact extraction systems, being NLP-driven, might be able to extract more information, and in a format more amenable for using in NLP applications than

[2]While CYC had a very broad set of applications in mind, NLP, Machine Translation, and Information Retrieval were some of the key motivations, as stated in [78]

manually curated resources. However, using knowledge-lean NLP tools to extract knowledge unavoidably results in noisy knowledge. For example, [127] reports that their basic fact extraction system induced that *Cananada is-a city* from the sentence *"Toronto, Canada and other cities."*. The large-scale fact extraction systems are capable to a certain extent of taking advantage of redundancy in the data to prune out incorrect facts. However, the pruning process is typically knowledge-lean, and cannot differentiate between pruning out the incorrect *city* sense for *Canada* and legitimately ambiguous terms, such as *New York* being both *city* and *state*. When it comes to minority senses, such as *Canada* being the *Canadian hockey team*, the situation only gets worse. Therefore, it is clear that high quality knowledge extraction requires good NLP tools, which in turn require knowledge, leading essentially to a chicken and egg problem.

Instead of focusing on *obtaining* a large set of logic axioms, we are focusing on *using* knowledge-rich features in NLP solutions. In this thesis, we argue that knowledge and NLP are intertwined. Meaning, knowledge can improve the performance of NLP systems, but at the same time, a better NLP system can improve our ability to use and inject knowledge. We exemplify this co-dependence in our work by showing how raw text and knowledge from Wikipedia can be exploited for the task of named entity recognition, and how a better named entity recognition system can benefit concept resolution against Wikipedia, which injects further knowledge into raw text. We then use this knowledge to improve a state of the art co-reference system. This process is summarized in Figure 1.2.

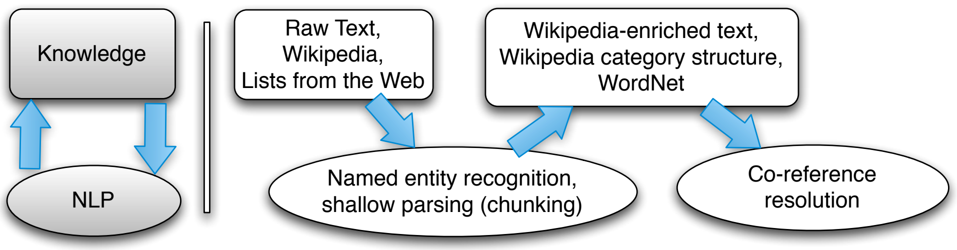

Knowledge
NLP
Raw Text, Wikipedia, Lists from the Web
Wikipedia-enriched text, Wikipedia category structure, WordNet
Named entity recognition, shallow parsing (chunking)
Co-reference resolution

Figure 1.2: The co-dependency between knowledge and NLP throughout this Thesis.

1.3 Contributions

Understanding natural language often involves answering five types of "Wh" questions: who, what, where, when and why. While full-blown text understanding is out of scope of this work, we focus on the first three questions (who, what, and where), which seem to be of a particular importance. Without knowing what are the entities in the text, it is impossible to understand the relations and the events that happen between them. We tackle the problem of answering the "wh" questions in text as a series of four increasingly complex NLP tasks described below. We have used knowledge extensively and developed state-of-the-art systems for each task.

1.3.1 Text categorization

Text categorization is the task of categorizing documents by topics, such as: politics, sports, medicine or science. Traditionally, the category names are ignored and treated as atomic identifiers. Hence, the task has been studied as the problem of training a classifier using labeled data. However, people can categorize documents into named categories without any explicit training because we know the meaning of category names. In [21], we have introduced *Dataless Classification*, a learning protocol that uses world knowledge to induce classifiers without the need for any labeled data. Like humans, a dataless classifier interprets a string of words as a set of semantic concepts. We proposed a model for dataless classification and showed that the label name alone is often sufficient to induce classifiers. Using Wikipedia as our (encyclopedic) source of knowledge, we have achieved above 85% accuracy on multiple classification datasets without *any labeled or unlabeled* data. When using unlabeled data, we could further improve the results and show a performance which is quite competitive to a supervised learning algorithm trained on 100 labeled examples.

1.3.2 Information extraction

Information extraction is the task of extracting pieces of relevant information from unstructured text. In [20] we have presented *CODL*, a framework for guiding semi-supervised learning with constraints. We have applied the framework to two problems: the task of extracting fields from apartment rental advertisements and the task of extracting the fields from scientific citations[3]. At the time the work was published, it achieved state-of-the-art results on both datasets. CODL uses constraints as a common-sense knowledge, and it will be the only instance of using non-encyclopedic knowledge throughout this thesis.

In [112] and in [145] we have tackled an important instance of information extraction problem known as named entity recognition, where the task is to identify locations, organizations and people in text. We have investigated system design challenges, techniques to capture non-local dependencies in text, and techniques to abstract over expressions using Wikipedia and raw text. At the time of its publication [112] achieved the best published result on CoNLL03 shared task dataset, the de-facto benchmark for the task.

[3]In the apartment rentals ads, the fields included the location, the number of bedrooms and the pets restrictions etc., and in the scientific citations, the fields were the author, the title, the publisher, the editors, the location of the conference, etc.

1.3.3 Concept Disambiguation

In concept disambiguation we cross-link expressions in the input free-form text to a database of concepts and entities. When the disambiguation is done against Wikipedia, we call this process *disambiguation to Wikipedia* or simply *Wikification* (a term coined by [90]). Throughout this work, we will use these terms interchangeably. Wikipedia is an attractive choice for reference collection dataset, since it is extremely comprehensive, and is constantly updated by the human editors. For example, "Michael Jordan" has a Wikipedia disambiguation page[4] providing 7 disambiguation candidates in the 2010 version of Wikipedia. Choosing the correct person in a context-sensitive way is challenging.

In [111] we have analyzed the approaches for leveraging structural information in Wikipedia (which we call "global" approaches) and compare them to more traditional (local) approaches. We showed that previous approaches for global disambiguation can be improved, but even then the local disambiguation provides a baseline which is very hard to beat. We presented the *GLOW* system which uses a unified optimization function to make inference over local and global features. GLOW achieved a new state-of-the-art Wikification performance on benchmark datasets and took the fourth place in the TAC 2011 entity-linking competition[5].

1.3.4 Co-reference Resolution

The co-reference resolution problem is the task of identifying expressions in text that refer to the same real-world entity. For example, in the text below, we want know that mentions $\{m1, m2, m4\}$ refer to Barack Obama and mentions $\{m3, m5\}$ refer to the American nation.:

> *"[[U.S. President]$_{\{m1\}}$ Barack Obama]$_{\{m2\}}$ spoke to the [nation]$_{\{m3\}}$. '[I]$_{\{m4\}}$ need [your]$_{\{m5\}}$ help', he said. . . . "*

In Chapter7 we are using GLOW to inject Wikipedia knowledge into a state-of-the-art co-reference resolution system of [10]. The key idea is to use Wikipedia to build a metric for measuring "compatibility" between two expressions. The main ingredients of the metric are nationality, gender and fine-grained typing of expressions facilitated by GLOW.

We note that one of the key reasons for superior performance of GLOW is that is uses a superior NER system of [112]. While benefitting from a superior NLP system, GLOW is used to inject better word knowledge into a co-reference resolution system of [10]. This co-dependence of knowledge and NLP is the main theme of this thesis, as illustrated in Figure 1.2.

[4]http://en.wikipedia.org/wiki/Michael_Jordan_(disambiguation).

[5]`http://nlp.cs.qc.cuny.edu/kbp/2011`

1.3.5 Using Knowledge in NLP

While the previous discussion has focused on the contribution of this thesis for advancing the state of the art for several applications, we also note that this thesis makes contributions in developing intuitions of how to use knowledge in NLP. We have experimented with several resources of knowledge and several methodologies of injecting knowledge into NLP applications. We have researched:

1. Data sparsity reduction methods based on using large quantities of unlabeled text as knowledge (Chapters 5 and 4).
2. Enriching text via encyclopedic knowledge. This can be done either by assigning topic-specific information (Chapter 3) or by cross-linking expressions in text to encyclopedic resources (Chapter 7).
3. Using first-order-logic-like constraints to guide self-training of a statistical model (Chapter 8).

1.4 Thesis Outline

The outline of this thesis closely follows the list of the contributions we have enumerated in the previous section. In Chapter 2 we cover technical background. In Chapter 3 we discuss how Wikipedia can assist with the task of texts categorization. In Chapter 4 we analyze the task of named entity recognition. We discuss the challenges and the main misconceptions of system design and proceed to focus on using unlabeled text and Wikipedia as means to inject knowledge into the system. In Chapter 5 we analyze methods to address data sparsity problem in NLP. In Chapter 6 we discuss concept disambigution to Wikipedia. In this case, rather than using knowledge to improve the system, we are building a system which will enrich input text with semantic knowledge. We show that better NLP tools, such as named entity recognition and shallow parsing (which benefited from knowledge derived from Wikipedia) allow us to inject higher quality information from Wikipedia to text. In Chapter 7, we are using the injected knowledge to improve a state-of-the-art co-reference system of [10]. While this thesis focuses on encyclopedic knowledge in NLP, we have experimented with several non-encyclopedic directions which will be discussed in Chapter 8. We conclude in Chapter 9.

Chapter 2

Background

Throughout this work we will heavily rely on techniques from machine learning for making predictions. A good example illustrating the nature of machine learning can be found in [38], who are considering a fish-packing plant which needs to automatically sort the fish to bass and salmon. The first step is taking measurements of the fish, for example the weight and the length. This process is known as *"feature extraction"*. The second step is gathering statistics for the two types of fish and learning a function to discriminate the fish with the highest possible accuracy given the limited information about the fish. The decision function may be simple or expressive, and generally, the more expressive functions will require more data to learn, but will eventually be more accurate when presented sufficient amount of training data [151; 152; 98].

Clearly, based on the two measurements of weight and length, it will be impossible to sort the fish with 100% accuracy. To achieve a good performance on the fish-sorting task, we need good measurements, which can be potentially expensive to obtain. Understanding natural language is tremendously more complex than sorting fish, and the key challenge (as is the case in most machine learning applications) is feature extraction. Text is composed of words, and words are currently the only type of features we can reliably extract, but the same ideas can be expressed in different words. This leads to data sparsity problems [68; 64] and other challenges which will be discussed later in great detail. Nevertheless, we can already outline the unique machine learning challenge of NLP: the decision functions often depend on a large vocabulary of words, but when the input text is presented, only a small fraction of the vocabulary is observed. These properties have led to the popularity of *linear models* in NLP.

2.1 Supervised Classification with Linear Models

Linear models are formalized as follows: given an input x, a possible output y and a feature extraction function ϕ, the goal is to learn a weight vector w, such that

$$w \cdot \phi(x, y^*) > w \cdot \phi(x, y^{'}) \tag{2.1}$$

where y^* is the correct output and y' is any other possible output. For example, in the "fish-sorting" example, the feature extraction function ϕ for a fish x^0 may look as follows:

$$\phi(x^0, bass) = (0, 0, length(x^0), weight(x^0))$$

and

$$\phi(x^0, salmon) = (length(x^0), weight(x^0), 0, 0)$$

In practice, we will have an additional feature set to value 1 for bias, and we can use squares of the measurements and the product of weight and length as features, but we omit these details for simplicity. A learned weight $w = (1, 3, 2, 1)$ would mean that if the ratio of length to weight in a fish is larger than $2 : 1$, then it will be classified as bass. To see why, consider the product $w \cdot \phi(x, y)$ for bass and salmon of a fish x^0 of length 2 and a weight 1. For bass $w \cdot \phi(x^0, bass) = (1, 3, 2, 1) \cdot (0, 0, 2, 1) = 5$, while for salmon $w \cdot \phi(x^0, salmon) = (1, 3, 2, 1) \cdot (2, 1, 0, 0) = 5$. This means that the fish x^0 is on the decision boundary. A fish of the same length, but a higher weight would be labeled as a bass, because the weight multiplier for bass is 3, and for salmon it is 1. Similarly, a fish of the same weight but a higher length would be classified as salmon. This framework can be replicated for NLP, with typically boolean features indicating presence or absence of words. We note that the linear models are a very large family including Naive Bayes, SVM, Perceptron, Winnow, Linear Regression, Max-Entropy classifiers [38; 19], etc. We also note that Kernel methods allow to learn linear decision boundaries over non-linear transformations of input feature vectors for many types of linear classifiers [71].

2.2 Structured and Sequential Prediction

A closer look at the modeling decision in Section 2.1 shows that each weight is replicated as many times as there are classes. With binary classification, this is not an issue, but when the number of target classes becomes very large, this approach becomes intractable. In NLP, we often deal with problems where the output space is very large. For example, consider the task of assigning a part of speech tag to every word in a sentence, e.g. *"NN/Smoking VBZ/is VBN/forbidden"*. Each word can be assigned over 40 tags, and with N words in a sentence, there are 40^N possible outputs. Clearly, replicating weights for each possible tagging of the sentence does not scale.

Luckily, natural language has structure, which allows us to use the *structured prediction* formalism. The key goal in structured prediction is to decompose the solution to inter-dependent parts and to capture the dependencies between the parts through features which allow only globally reasonable solutions. In this work, the structure under consideration will be sequential (as in the part of speech tagging, and in contrast to dependency parsing where the structures are tress.).

In structured prediction, given an input $\mathbf{x} = (x_1, \dots x_n)$, the task is to find the best assignment $\mathbf{y} = (y_1, \dots y_n)$ to the output variables. We denote $\mathcal{X}$ to be the space of the possible inputs and $\mathcal{Y}$ to be the set of possible outputs. We define a structured output classifier as a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ which uses a global scoring function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ to assign scores to each possible (x, y) pair. Given an input $\mathbf{x}$, a desired function f will assign the highest score to the correct output $\mathbf{y}$ among all the possible outputs. The global scoring function is often decomposed as a weighted sum of feature functions,

$$f(x, y) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}) \tag{2.2}$$

Note the similarity of Equation 2.1 and Equation 2.2. The "only" difference is that the latter operates on vectors (or other structures) rather than on atomic instances. That is, the key attribute of structured prediction is the ability to decompose the output into a set of interdependent structures. This allows to add dependencies between the different components of $\mathbf{y}$. Therefore, it seems natural that many "simple" linear classifier formalisms have their "structured" twin, such as structured SVM [144] and structured perceptron [25].

However, it is important to note that learning and inference in the structured scenario become challenging. For example, it is possible to learn the weight w jointly as is done in [25], or to reconstruct the structured output component-by-component through sequentially applying multi-class classifiers as was done in [121]. Similarly, the search of the optimal structure y for input x can be done through dynamic programming [107], beam search [20], or integer liear programming [149]. Moreover, it is beneficial to couple the training and the inference [32].

2.3 NLP-specific Considerations

In Sections 2.1 and 2.2 we have discussed linear models, optionally with capability to handle structure. There are several instances of these models which are very popular in NLP and deserve a special discussion. In this section, we briefly discuss NLP-specific models and concepts. Consider the sentence *"Michael Jordan was the best player in the NBA"*. We might ask several questions about this sentence:

1) Is this sentence about basketball or baseball?

2) What people, places, organizations etc. are mentioned in the text?

Below, we discuss ways to represent text in a way which can be used in Machine Learning.

2.3.1 Modeling Text as a Bag of Words

The task of deciding whether *"Michael Jordan was the best player in the NBA"* discusses baseball or basketball is known as text categorization. The simplest approach (which works surprisingly well) is to map each word in a vocabulary of size N to a symbolic feature ID. Sentences can be represented as N-dimensional vectors, there the i-th component represents presence or absence of the i-th word of the vocabulary in the input text. This text representation is known as bag of words (BOW). The task of text categorization becomes the task of assigning labels to feature vectors with two main research questions: what function to learn over the vectors and what algorithm to use to learn that function? Multiple frameworks have been proposed, such as Naive Bayes, Perceptron, Logistic Regression, SVM etc. Surprisingly, all the above frameworks can be shown to learn a linear classifier over a possible transformation of the feature vectors with different loss functions [118; 119].

It is important to note that often we will prefer to use real-valued vectors, where the i-th entry will indicate the relative importance of the i-th word, rather than its presence. TF-IDF is an important weighting scheme, which is often used for this purpose. TF-IDF of a word w in text t is defined as $\frac{c(w,t)/|t|}{log(|D|/|\{j:w\in d_j\}|)}$, where $c(w,t)$ is the number of times w appears in t, D is a document collection, $|D|$ is the size of the document collection, and $c(w,t)$ is the number of documents in which w appears. Intuitively, a word is descriptive of text t if it appears a lot in t (has high term frequency (TF)), but is rare in the other documents (has a low inverse document frequency (IDF)). Note that the IDF scores are computed with respect to some document collection. When we categorize an input text, it is sometimes given as a part of a large document collection, but sometimes this is not the case. In these cases, we can use the Wikipedia as a reference document collection.

2.3.2 Modeling Text as a Sequence of Words

BOW representation ignores the order of words in the text. However, in structured prediction tasks, the order of words may be crucial. For example, consider the task of detecting named entities in the following sentence. *"y_0/Michael y_1/Jordan y_2/was y_3/the y_4/best y_5/player y_6/in y_7/the y_8/NBA"*. Typically this is modeled as the task of predicting the tags $(y_0, \ldots, y_8)$ in a way which allows to reconstruct the boundary and the type of the text segments. While multiple text chunk segmentation schemes have been developed [85; 156; 129], we show in [112] that the BILOU tagging scheme leads to the best performance in named entity recognition (NER). In the BILOU tagging scheme, B marks the beginning of a chunk, L marks the last token of the chunk, I marks a token inside of a chunk, U indicates a unit-token segment, and O indicates a non-entity.

With BILOU encoding, the correct NER tags for our sample sentence would be: *"B-PER/Michael L-PER/Jordan O/was O/the O/best O/player O/in O/the U-ORG/NBA"*. From this tagging, it is possible to induce that *Michael Jordan* is a person and *NBA* is an organization. Note that most tagging schemes introduce dependencies between the tags. For example, in the NER task, the tag *B-PER* can only be followed by *I-PER* or *L-PER*; the tags *I-ORG*, *U-PER* or *O* are not allowed. Therefore the feature extractor $\Phi(\mathbf{x}, \mathbf{y})$ of Equations 2.2, in addition to the obvious (y_i, x_i), extracts features over the pairs (y_i, y_{i+1})[1]. Several models have been suggested for solving the problem, such as hidden Markov models [107], Markov random fields, structured preceptron [25], structured support vector machines [144], conditional random fields [75], max entropy Markov models [105], max margin Markov models [137] etc. Interestingly, they can all be viewed as structured linear models [119; 25].

2.4 Learning Protocols

2.4.1 Supervised Learning

The most common scenario in machine learning is when we are given a labeled training set L from which we learn our model (the model is the weight vector w of a linear classifier in Equation 2.1). This training protocol is called *supervised learning*. However, supervised learning in NLP has several limitations stemming from the data sparsity problem. First, a lot of data is typically necessary for good generalization. Second, language idiosyncrasies of the training domain can create domain adaptation issues [34; 65; 31]. These limitations of supervised learning on knowledge-lean word-based features often call us for using unlabeled data. Much of the work in this thesis is directed towards addressing the data sparsity problem in NLP. In Chapter 3 we show how to remove the need for the labeled data in text classification and in Chapter 4 we show how to use knowledge-rich features to address the problem of domain adaptation in named entity recognition.

2.4.2 Semi-supervised Learning

Semi-supervised learning is a learning protocol where unlabeled data is used to supplement the labeled data to train a model. Several semi-supervised training algorithms have been proposed. Most notably: self training (aka bootstrapping) [166], co-training [13], alternating structure optimization [1], and various flavors of EM [35; 89; 11]. It is important to note that using unlabeled data in most aforementioned approaches is

[1]Other pairs and triplets can be features as well.

not straightforward. In Chapter 4 we demonstrate that it is possible to induce from unlabeled data word representations which can be computed once and may be used for a variety of tasks.

2.4.3 Unsupervised Learning

Sometimes, the number of target labels is not known or is not even defined. For example, consider the co-reference resolution task of *clustering* all the mentions that refer to the same entity into a single cluster. Similarly, consider grouping of words by meaning at some granularity level. In these scenarios, a *similarity metric* is used as an input to a clustering algorithm. In this thesis, we will consider scenarios when during testing we encounter new words which were not observed during training. In these cases, if we have a good similarity metric, we can still induce the properties of the unfamiliar word by considering similar words which we have already observed during the training. The *distributional similarity hypothesis* [59] states that words which appear in the similar contexts carry similar meaning. The approaches for semi-supervised training discussed in Chapters 4 and 5 are built based on the distributional similarity hypothesis, which is in turn, an unsupervised approach. The "dataless" approach we discuss in section Chapter 3 is also an unsupervised approach.

Chapter 3

Dataless Classification

In this section we start investigating the "who" and "what" questions by looking at the text categorization task, which is traditionally seen as the problem of assigning a label to data. Often, semantic information available in the label is ignored while making this decision and the labels are treated as atomic identifiers. This necessitates the use of annotated data to train classifiers that map documents to these identifiers. On the other hand, humans can perform text categorization without seeing even one training example. For example, consider the task of deciding whether a Usenet post must be posted to *comp.sys.mac.os* or *sci.electronics*. We can do this without explicit training because we use the meaning of the labels to categorize documents.

In this section, we introduce the notion of *Dataless Classification* – a model of classification that does not need annotated training data. Our approach is based on the use of a source of world knowledge to analyze both labels and documents from a semantic point of view, allowing us to learn classifiers. Such analysis by a semantic interpreter allows us to compare the *concepts* discussed by the document to perform classification.

Using a semantic knowledge source which is based on Wikipedia, we experimentally demonstrate that dataless classification can categorize text without any annotated data. We show the results of classification on the standard 20 Newsgroups dataset and a new Yahoo! Answers dataset. Without even looking at unlabeled instances, dataless classification outperforms supervised methods. Moreover, when *unlabeled* instances are available during training, our methods are comparable to the supervised methods that need 100 training examples.

We can perform *on the fly* text categorization for previously unseen labels, since our classifier was not trained on any particular labels. Furthermore, since we do not need previously labeled data, the model is not committed into any particular domain. Therefore, we can use dataless classification across different data sets. We experimentally show that our method works equally well across domains.

3.1 Semantic Representation

Traditionally, in most text categorization tasks, labels are treated as atomic identifiers, thereby losing their meaning. The task of text categorization is regarded as the task of learning a classifier that can distinguish between two abstract categories represented by these identifiers. However, labels often contain valuable semantic information that could be utilized for learning the classifiers.

The simplest approach to represent the semantics of a text fragment is to treat it as a vector in the space of words. We refer to this as the *bag of words (BOW)* representation. While the bag of words representation is one of the most commonly used representation for documents, documents *and* labels have semantic content which often goes beyond the words they contain. For example, though the phrase *'American politics'* can be treated as just the two words, it could connote discussion about a wide range of topics – Democrats, Republicans, abortion, taxes, homosexuality, guns, etc. This indicates that a text fragment can be treated as a vector in the space of concepts, if such a representation is available. The vector of concepts could be thought of as the semantic interpretation of the text fragment.

In order to obtain a more meaningful semantic interpretation of a text fragment, we use *Explicit Semantic Analysis (ESA)*, that was introduced in [50] and uses Wikipedia as its source of world knowledge. ESA was originally introduced to measure semantic relatedness between text fragments. Given a text fragment, the ESA algorithm generates a set of concepts that are weighted and ordered by their relevance to the input. Here, we provide a brief summary of this approach and refer the reader to [50] and [51] for more details.

The main assumption is that each article in Wikipedia corresponds to a concept. To get the ESA representation of a text fragment, for each word in the text, the interpreter identifies the concepts that contain it. These concepts are combined to form a weighted vector, where the weights are obtained by using the TFIDF representation of the original text. The list of concepts are ordered by the weight to get the final ESA representation.

Since Wikipedia is the largest encyclopedic source of knowledge on the web, ESA representation is sufficient for many categorization tasks. Additionally, since Wikipedia was generated by humans, it provides a natural measure of relatedness between text fragments. Previous research has shown that semantic interpretation based on Wikipedia is a more reliable measure of distance between documents than the traditional bag-of-words approach. In the Discussion section, we present brief thoughts on why dataless classification works and what makes a resource of knowledge sufficient for this purpose.

In our experiments, we use the BOW and ESA representation schemes for both label names and the documents.

3.2 Data and Experimental Methodology

To evaluate our ideas, we need datasets where the labels contain rich semantic information. Here, we describe the two datasets that we used.

3.2.1 20 Newsgroups Dataset

The 20 Newsgroups Dataset is a common benchmark used for testing classification algorithms. The dataset, introduced in [77], contains approximately 20,000 newsgroup posts, partitioned (nearly) evenly across 20 different newsgroups. Some of the newsgroups are very closely related to each other (e.g. *comp.sys.ibm.pc.hardware* and *comp.sys.mac.hardware*), while others are unrelated (for example, *misc.forsale* and *soc.religion.christian*). The 20 topics are organized into broader categories: computers, recreation, religion, science, forsale and politics. This dataset has been used for many learning tasks [110; 88].

Since our approach uses the content of label names, good label names are essential for the performance. We cleaned the existing labels by expanding the newsgroup names that were used in the original data. For example, we expanded *os* into *operating system* and *mac* to *macintosh apple*. We also removed irrelevant words – *misc, alt, rec* and *talk*. The amended label names given for each class are summarized in Table 3.1. This amendment is necessary because the existing labels are sometimes not real words or, as in the case of *for sale*, the expansion contains a stop word, which necessitates further expansion.

Yahoo! Answers

The second dataset that we used for our experiments is based on Yahoo! Answers and was collected for these tasks.[1] We extracted 189,467 question and answer pairs from 20 top-level categories from the Yahoo! Answers website (that is, about 10,000 question/answer pairs per category). These top-level categories have a total of 280 subcategories which refine the labels, though the distribution of question/answer pairs at the subcategory is not uniform. Table 3.2 shows a sample of category and subcategory names. For our experiments, we used the original category and subcategory names as label names.

3.2.2 Experimental Methodology

[110] used the 20 Newsgroup dataset to construct ten binary classification problems, shown in Table 3.3. We used these problems for our experiments. In [110], they use a logistic regression classifier. As our

[1]Please contact the authors for the Yahoo! Answers data.

Newsgroup Name	**Expanded Label**
talk.politics.guns	politics guns
talk.politics.mideast	politics mideast
talk.politics.misc	politics
alt.atheism	atheism
soc.religion.christian	society religion christianity christian
talk.religion.misc	religion
comp.sys.ibm.pc.hardware	computer systems ibm pc hardware
comp.sys.mac.hardware	computer systems mac macintosh apple hardware
sci.electronics	science electronics
comp.graphics	computer graphics
comp.windows.x	computer windows x windowsx
comp.os.ms-windows.misc	computer os operating system microsoft windows
misc.forsale	for sale discount
rec.autos	cars
rec.motorcycles	motorcycles
rec.sport.baseball	baseball
rec.sport.hockey	hockey
sci.crypt	science cryptography
sci.med	science medicine
sci.space	science space

Table 3.1: Newsgroup label names. We expanded the names of the newsgroups to full words and removed some words like misc. This table lists the expanded newsgroup names.

supervised baseline, we trained a naive Bayes classifier using 10 labeled examples and observed similar accuracies.

For the Yahoo! Answers dataset, we generated 20 random binary classification problems at the subcategory level. Some of these problems are shown in 3.4. We report the average performance over all the problems and then focus on specific cases which highlight the strengths and weaknesses of our approach.

Intuitively, these are 'easy' classification problems for humans. However, with 10 training samples, [110] reported the error rates as high as 20% in 8 out of 10 problems, and even with 100 labeled samples, the error rate on the *religion vs. politics.guns* problem was above 20%. Using the on-the-fly classification technique described in the next section, we achieve error rates below 11.5% on 9 out of 10 problems with *no* labeled data at all.

Top-level Category	**Subcategory**
Arts And Humanities	Theater Acting
Business And Finance	Advertising Marketing
Business And Finance	Taxes
Computers And Internet	Security
Consumer Electronics	Play Station
Entertainment And Music	Jokes Riddles
Games And Recreation	Video Online Games
Sports	German Football Soccer
Sports	Rugby League

Table 3.2: Sample Yahoo! Answers categories and subcategories. In all, we collected 20 top level categories and 280 subcategories from Yahoo! Answers.

Id	**Problem**
1	Motorcycles Vs Ms-Windows
2	Baseball Vs Politics.misc
3	Religion Vs Politics.guns
4	Atheism Vs Autos
5	IBM hardware Vs Forsale
6	Politics.mideast Vs Sci.med
7	Christianity Vs Hockey
8	Space Vs Mac.Hardware
9	Windows.X Vs Electronics
10	Sci.Cryptography Vs Comp.graphics

Table 3.3: The set of 10 binary classification problems used in [110] for the 20 newsgroups data.

3.3 On-the-fly Classification

3.3.1 Classification with *no* data

We use the term "on-the-fly classification" to refer to the situation where category names are not known in advance and hence, no training can be done. Essentially, on-the-fly classification is a technique for classifier induction that does not use *any* data at all and uses no knowledge of the datasets either. We show that with no data other than the label names, we can get very good performance for classifying documents. The efficacy of this approach stems from the rich semantic representation discussed earlier. In order to demonstrate the effectiveness of the semantic representation, we use a very simple classification technique – we perform Nearest Neighbors classification on an appropriately selected feature space. Let $\varphi(t)$ denotes some vector representation of text t. For a document d and labels l_i, we pick the category i if $arg\min_i ||\varphi(l_i) - \varphi(d)||$.

Using the Bag of Words (BOW) representation, which represents text fragments as vectors in the space of words, we get the **NN-BOW** classifier. The ESA representation represents text as vectors in the space of concepts. Using the ESA representation gives us the **NN-ESA** classifier.

Id	Description
⋮	⋮
14	Health Diet Fitness Health Allergies
15	Business And Finance Small Business Consumer Electronics Other Electronics
16	Consumer Electronics DVRs Pets Rodents
17	Business And Finance India Business And Finance Financial Services
18	Sports Mexican Football Soccer Social Science Dream Interpretation
19	Sports Scottish Football Soccer Pets Horses
20	Health Injuries Sports Brazilian Football Soccer

Table 3.4: Sample binary categorization problems for the Yahoo! Answers dataset – subcategory level

Dataset	Supervised Baseline	NN-BOW	NN-ESA
Newsgroups	71.71	65.73	85.29
Yahoo	84.34	66.79	88.62

Table 3.5: Average accuracy of dataless classification with the bag of words and ESA representations on the Newsgroups and Yahoo! Answers datasets. Note that in both domains, the use of ESA representation is better than using just the words of the labels and no other training data. In fact, the ESA representation, without any training data outperforms supervised naive Bayes classifiers which use ten labeled examples.

3.3.2 Results

The results of dataless classification on the binary categorization tasks for the Newsgroups and the Yahoo! Answers datasets are summarized in Table 3.5. It is clear that the **NN-BOW** algorithm can classify documents correctly only if the document contains words that appeared in the label names. Therefore, the recall of this method is quite limited. We can see **NN-ESA** performs much better than bags of words approach. This implies **NN-ESA** can categorize of documents even if they share no common words with the label name. Since the ESA representation maps a text snippet into high-dimensional semantic space, two documents may be very close in the ESA space even if they share no common terms. This results in an on-the-fly classifier that can use dynamic, ad-hoc labels.

In Table 3.5, we also show as baseline, the performance of a supervised naive Bayes classifier that was learned with ten examples. We can see that with both the Newsgroups and Yahoo! Answers datasets, the **NN-ESA** classifier outperforms the supervised classifier.

3.4 Semantics of Unlabeled Data

3.4.1 Modeling Unlabeled Data

In the previous section, we show that even without knowing anything about the data, a rich semantic representation can perform on par with supervised methods. However, often we have access to unlabeled documents belonging to the problem of interest. This allows us to take advantage of previous work in semi-supervised learning (Refer, for example, [101]).

Bootstrapping

A straightforward extension of the on-the-fly scheme is to bootstrap the learning process with only the label names as examples. Algorithm 1 presents the pseudo-code for learning a bootstrapped semi-supervised classifier using a feature representation φ. Note that though we do perform training, we still do not use any labeled data and use only the label names as the starting point in the training. (Steps 1 to 4 indicate this.) Using the BOW and ESA representations with this algorithm gives us the **BOOT-BOW** and **BOOT-ESA** classifiers respectively.

Algorithm 1 Bootstrap-φ. *Training a bootstrapped classifier for a feature representation φ, where φ could be Bag of Words or ESA.*

```
1:  Let training set $T = \emptyset$
2:  for all labels $l_i$ do
3:    Add $l_i$ to $T$ with label $i$
4:  end for
5:  repeat
6:    Train a naive Bayes classifier $NB$ on $T$
7:    for all $d_i$, a document in the document collection do
8:      If $y = NB.classify(\varphi(d_i))$ with high confidence
9:      Add $d_i$ to $T$ with label $y$
10:   end for
11: until No new training documents are added.
```

Co-training

The classifiers **Boot-BOW** and **Boot-ESA** are learned by bootstrapping on the bag of words and the ESA representations of the data respectively. The fact that BOW and ESA are parallel representation of the same data is ignored. Prior work ([13]) has studied the scenario when two independent feature representations (or views, as they are called in [13]) $\varphi_1(d)$ and $\varphi_2(d)$ are available for the same data and that if each feature representation is sufficient for correct classification of the data. In such a case, we can train two classifiers

Dataset	Supervised 10 Samples	Supervised 100 Samples	BOOT-BOW	BOOT-ESA	Co-train
Newsgroup	71.71	92.41	88.84	90.92	92.70
Yahoo! Answers	84.34	94.37	90.70	92.73	95.30

Table 3.6: Performance of classifiers when unlabeled data is available. The supervised baseline, in this case, is a naive Bayes classifier that is trained on 100 labeled examples. The last three classifiers do not use any labeled examples at all.

$c_{\{\varphi_1\}}$ and $c_{\{\varphi_2\}}$ that classify data better than chance. These two classifiers can train one another in a procedure called Co-training. We can apply a variant of this idea for our task. The algorithm for co-training is summarized in Algorithm 2. While the ESA representation is a function of the BOW representation, violating the 'view independence' assumption, we show that in practice, this procedure leads to a satisfactory performance. [2]

Algorithm 2 Co-training *We use the fact that BOW and ESA can independently classify the data quite well to induce a new classifier.*

```
1: Let training set $T^{BOW} = \emptyset$,$T^{ESA} = \emptyset$.
2: **for all** labels $l_i$ **do**
3:     Add $l_i$ to both $T^{ESA}$ and $T^{BOW}$ with label $i$
4: **end for**
5: **repeat**
6:     Train a naive Bayes classifier $NB^{BOW}$ on $T^{BOW}$.
7:     Train a naive Bayes classifier $NB^{ESA}$ on $T^{ESA}$.
8:     **for all** $d_i$, a document in the document collection **do**
9:         **if** Both $NB^{BOW}$ and $NB^{ESA}$ classify $d_i$ with high confidence **then**
10:            Add $d_i$ to $T^{BOW}$ with label from $NB^{BOW}$
11:            Add $d_i$ to $T^{ESA}$ with label from $NB^{ESA}$
12:        **end if**
13:    **end for**
14: **until** No new training documents are added
```

3.4.2 Results

The performance of the different algorithms with the Newsgroup and Yahoo! Answers datasets is summarized in Table 3.6. Additionally, we also show the performance of a supervised baseline system that used a naive Bayes algorithm with 100 training examples. We can see from the table that both the **BOOT-ESA** and **Co-train** classifiers are competitive with the supervised classifier. Even the **BOOT-BOW** classifier is comparable to the supervised baseline. This indicates that using unlabeled data, we can train very strong classifiers if we use the semantics of the labels.

[2]We also experimented with concatenating the BOW and the ESA representations into a single BOW+ESA view as was done in [49] for supervised classification. Then we bootstrapped the naive Bayes classifier on the BOW+ESA view, but it consistently performed worse than co-training.

3.5 Discussion

Our results have showed that using a good representation can have a dramatic impact on classification. In this section, we discuss when and why simply using the label name is sufficient for classification and provide a justification for our method. Our goal is to develop an understanding of what makes a dataset a good semantic resource, so that it will be possible to apply our framework using resources other than Wikipedia.

Let W be the set of all words. Consider a classification task in which the categories are $c_1, c_2, \ldots, c_m$. We assume that there exist collections $G_1, G_2, \ldots, G_m$ of words that are 'good' for these categories. Informally, we interpret 'good' to have a discriminative meaning here. That is, a word is 'good' for c_i if it is more likely to appear in c_i-documents than in documents of other categories.

The notion of 'good' directly relates to a key assumption about the relation between category labels and the semantic knowledge resource that we use (in our experiments, Wikipedia). Specifically, if a user chooses l to be a label of a category c, and thus thinks that l is a 'good description' of the category, it means that Wikipedia articles that contain l should contain other words that are 'good' for c. Consequently, our approach for generating a semantic representation using l and the Wikipedia articles will bring a document that belongs to category c closer to the representation of l, than it would a document that does not belong to c. Even if the approximation by the label name is not of a high quality, given a good representation, classifiers can be discriminating.

Let $\varphi(.)$ be the semantic representation which receives a document and generates the semantic representation. First, we assume that the representation is good. This means that there exits a an oracle document w^i for category i such that all documents that belong to c_i are close to w^i. More formally,

$$\|w^i - \varphi(d)\| \leq \|w^j - \varphi(d)\| - \gamma, \forall j \neq i \tag{3.1}$$

where γ can be viewed as margin.

Note that our approximation for w^i is by $\varphi(\{l_i\})$ which is the representation for the label name of category c_i. We say that the approximation is reasonable, which means the distance between $\varphi(\{l_i\})$ and w^i is not too far. Formally,

$$\|w^i - \varphi(\{l_i\})\| < \eta.$$

By triangle inequality,

$$\begin{aligned} \|\varphi(d) - \varphi(\{l_i\})\| &\leq \|w^i - \varphi(d)\| + \|w^i - \varphi(\{l_i\})\| \\ &= \|\varphi(d) - w^i\| + \eta. \end{aligned} \tag{3.2}$$

Combining (3.1) and (3.2), it follows that

$$\begin{aligned} \|\varphi(d) - \varphi(\{l_i\})\| &\leq \|w^i - \varphi(d)\| + \eta \\ &\leq \|w^j - \varphi(d)\| - \gamma + \eta, \forall j \neq i, \\ &\leq \|\varphi(d) - \varphi(\{l_j\})\| - \gamma + 2\eta, \forall j \neq i \end{aligned}$$

The final step of the above inequality implies that if the representation is good (that is, if γ is large enough), then η can be as large as $\gamma/2$ without changing the classification result. This justifies the intuition that if the distance between the categories is large enough in some space, then the approximation that the labels provide for the categories need not be perfect for good classification performance.

The previous discussion suggests that often, with the appropriate representation, dataless classification is *easy* since the categories are far apart. We believe that, often, when the problem seems to be hard for dataless classification, it could be due to a very specific definition of a category, that may be different than the one expressed in our resources (Wikipedia).

Newsgroup name	**Discriminative words**
Christianity	morality, host, nntp, writes, koresh
Religon	thanks, quite, assume, clh, rutgers

Table 3.7: The 'discriminative words' found by labeled data of the task *Christianity vs. Religion* in 20 newsgroup.

Consider, for example, the difficult pair of *Christianity vs. Religion* in the 20 Newsgroup data. We have listed the top five discriminative words (that is, highest weighted words for both classes according to a discriminative classifier) for this task in Table 3.7. It is clear that (i) it is difficult to distinguish the document based on these words even for human, and (ii) there are some unrepresentative words which happened to be 'good' in this task. Since it is unlikely we can find a document in Wikipedia mentions *Christianity* and *nntp* at the same time, it is hard for dataless classification to perform well. Therefore, it is crucial to the success of the task is that the 'good' words for the specific classification task should be representative to their label names.

3.6 Dataless Classification for Domain Adaptation

The problem of discriminating two concepts across several domains is an important open problem in machine learning called *domain adaptation*, which recently has received a lot of attention [34; 65]. In this section, we claim that the semantic representation of documents are 'universal' and works across domains. Therefore, in document classification, universal representation diminishes the necessity of domain adaptation.

In traditional approaches to text categorization, when a classifier is trained on a given domain, it may not categorize documents well in a new domain. Informally, the classifier has learned the vocabulary used to express the category's documents in a specific domain because different domains might use different words to express the same ideas.

On the other hand, using the dataless approach simplifies the problem of domain adaptation. First of all, the label names are the same from different domains. Furthermore, documents of the same category from different domains should project to similar concepts because of the wide coverage of Wikipedia. Therefore, projecting the documents onto the space of concepts works well across domains. Our interpretation is that semantic representation categorizes documents as belonging to a category based on their *meaning* rather than the surface representation.

Test	BOW representation		ESA representation		
Data	Train on 20NG	Train on Yahoo	Train on 20NG	Train on Yahoo	Dataless
20 NG	0.97 → 0.60		0.96 → 0.90		0.96
Yahoo	0.89 ← 0.93		0.96 ← 0.97		0.94

Table 3.8: Analysis for adaptation from $Source \rightarrow Target$ for different domains.

As our running example, we choose to focus on discriminating documents pertaining to *baseball* and *hockey*. The categorization accuracy with 5-fold cross-validation are 0.97 for the Newsgroup domain, and 0.93 for the Yahoo domain using BOW approach (training data and testing data are from the same domain).

The results for domain adaptation are shown in Table 3.8. When we applied the NB classifier trained on the BOW representation of 20NG to Yahoo data , the accuracy dropped from 0.93 down to 0.89, and when we applied the classifier trained on Yahoo to the Newsgroup domain, the accuracy dropped down significantly from 0.97 to 0.60. This shows that the BOW classifiers are very sensitive to data distribution. In contrast, when ESA representation is used instead of BOW, 5-fold cross validation was 0.96 on 20NG and 0.97 for Yahoo. When the NB classifier trained on the ESA representation of Yahoo documents was applied to 20NG, the performance dropped only slightly to 0.90. When we applied the classifier trained on ESA representation of 20NG documents to Yahoo, the accuracy was 0.96.

However, the most significant result is that when we applied the dataless algorithm **NN-ESA**, presented in Section 3.3.1 (where we used only the label name), the performance was 0.94 on the 20NG dataset and 0.96 on the Yahoo dataset, which are very competitive with the result by supervised learning algorithm with *in-domain* training data.

These results demonstrate the good adaptation properties of the ESA-representation-based approaches in general, and the dataless NN-ESA approach presented in this paper, which uses the universal cross-domain semantic information present in the label to classify data across domains.

3.7 Conclusions and Future Work

Quite often, classification tasks are defined by specifying labels that carry meaning. Text categorization is a prime example in that the labels identify a category with words that describe the category. In this paper, we develop the notion of *Dataless Classification* that exploits this situation to produce on-the-fly classifiers without the need for training. Furthermore, using the unlabeled data, we can improve our training to get highly accurate classifiers without looking at any labeled examples.

We note that, unlike traditional learning, where we need at least two classes to learn a classifier, the idea of using a semantic interpreter could be applied to create a one-class classifier too. For example, we could think of a 'baseball-classifier', which identifies documents that discuss baseball. This bridges the notions of classification and information retrieval.

The semantic interpreter plays an important role in the performance of our system. An appealing aspect of ESA is that it covers a wide-ranging set of topics. However, we could replace Wikipedia with a different source of world knowledge and define our semantic interpreter using this source. For example, in [49], the authors use the Open Directory Project to construct a semantic interpreter. Additionally, we could create semantic interpreters with specialized data sources if we are interested in categorizing documents related to a specific area.

Chapter 4

Named Entity Recognition

In this section, we continue exploring the *"who, what," and "where"* questions by considering the task of extracting entities of interest from text. Named Entity Recognition (NER) is an information extraction task, where the goal is to identify people, locations and organizations in the text. Like many NLP applications, NER is characterized by making complex interdependent decisions that require large amounts of prior knowledge. Figure 4.1 shows a sample output of an NER system, illustrating some of the challenges in named entity recognition. In the absence of mixed case information it is difficult to understand that *"BLINKER"* is a person. Likewise, it is not obvious that the last mention of *"Wednesday"* is an organization (in fact, the first mention of *"Wednesday"* can also be understood as a "comeback" which happens on Wednesday). An NER system could take advantage of the fact that "blinker" is also mentioned later in the text as the easily identifiable "Reggie Blinker". It is also useful to know that *Udinese* is a soccer club (an entry about this club appears in Wikipedia), and the expression *"both Wednesday and Udinese"* implies that *"Wednesday"* and *"Udinese"* should be assigned the same label.

4.1 Contributions

Despite the recent progress in NER, the effort has been dispersed in several directions and there are no published attempts to compare or combine the recent advances, leading to some design misconceptions and less than optimal performance. In [112] we have analyzed some of the fundamental design challenges and

SOCCER - [PER BLINKER] BAN LIFTED .
[LOC LONDON] 1996-12-06 [MISC Dutch] forward [PER Reggie Blinker] had his indefinite suspension lifted by [ORG FIFA] on Friday and was set to make his [ORG Sheffield Wednesday] comeback against [ORG Liverpool] on Saturday . [PER Blinker] missed his club's last two games after [ORG FIFA] slapped a worldwide ban on him for appearing to sign contracts for both [ORG Wednesday] and [ORG Udinese] while he was playing for [ORG Feyenoord].

Figure 4.1: Example illustrating challenges in NER.

misconceptions that underlie the development of an efficient and robust NER system. We find that BILOU representation of text chunks significantly outperforms the widely adopted BIO. Surprisingly, naive greedy inference performs comparably to beamsearch or Viterbi, while being considerably more computationally efficient. We analyze several approaches for modeling non-local dependencies proposed in the literature and find that none of them clearly outperforms the others across several datasets. However, as we show, these contributions are, to a large extent, independent and, as we show, the approaches can be used together to yield better results.

One of the key contributions of [112] is corroborating the recently published results [91; 80; 72] indicating that word class models known as Brown Clusters can significantly improve the performance of NLP systems. Brown Clusters can be induced from unlabeled text and can be an alternative to traditional semi-supervised learning paradigm. In [145] we have explored additional approaches to represent words for alternative semi-supervised learning beyond Brown Clusters.

Combining recent advances, we develop a publicly available NER system[1] which achieves 90.8 F_1 score on the CoNLL-2003 NER shared task, among the best reported result for this dataset. Our system is robust – it consistently outperforms all publicly available NER systems (e.g., the Stanford NER system) on out of domain data.

4.2 Design Challenges in NER

In this section we introduce the baseline NER system, and raise the fundamental questions underlying robust and efficient design. NER is typically viewed as a sequential prediction problem, the typical models include HMM [107], CRF [75], and sequential application of Perceptron or Winnow [25]. That is, let $\mathbf{x} = (x_1, \ldots, x_N)$ be an input sequence and $\mathbf{y} = (y_1, \ldots, y_N)$ be the output sequence. The sequential prediction problem is to estimate the probabilities

$$P(y_i | x_{i-k} \ldots x_{i+l}, y_{i-m} \ldots y_{i-1}),$$

where k, l and m are small numbers to allow tractable inference and avoid overfitting. This conditional probability distribution is estimated in NER using the following baseline set of features [163]:

[1]http://cogcomp.cs.illinois.edu/page/software

1. Previous two predictions y_{i-1} and y_{i-2}
2. Current word x_i
3. x_i word type (all-capitalized, is-capitalized, all-digits, alphanumeric, etc.)
4. Prefixes and suffixes of x_i
5. Tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
6. Capitalization pattern in the window c
7. Conjunction of c and y_{i-1}.

Most NER systems use additional features, such as POS tags, shallow parsing information and gazetteers. We discuss additional features in the following sections. We note that we normalize dates and numbers, that is *12/3/2008* becomes **Date**, *1980* becomes **DDDD** and *212-325-4751* becomes **DDD*-*DDD*-*DDDD**. This allows a degree of abstraction to years, phone numbers, etc. We also use two types of tokenization simultaneously- we aggressively tokenize on hyphens and punctuation marks, but we also record hyphenated items as one entity, we also record full stops when we speculate it can be an abbreviation. That is, given a sentence:

CA.-based Google Inc. has purchased YouTube. we tokenize it as follows:

[CA . - based] Google [Inc .] has purchased YouTube .

where each expression in the square brackets will contribute several types of features. Due to space limitations, we do not provide the full details of the dual tokenization scheme. We notice that the main goal of the dual tokenization scheme is to make the system robust to the different tokenization schemes existing in CoNLL and out-of-domain data.

Our baseline NER system uses a regularized averaged perceptron [48]. Systems based on perceptron have been shown to be competitive in NER and text chunking [69; 105; 18] We specify the model and the features with the LBJ [117] modeling language. We now state the four fundamental design decisions in NER system which define the structure of this paper.

4.2.1 Datasets and Evaluation Methodology

In the following sections we present various approaches to address the questions posed in Figure 4.2 and evaluate their impact on system performance. Prior to doing that, we introduce the methodology we use for evaluating the system performance.

Key design decisions in an NER system.
1) How to represent text chunks in NER system?
2) What inference algorithm to use?
3) How to model non-local dependencies?
4) How to use external knowledge resources in NER?

Figure 4.2: Key design decisions in NER.

NER system should be robust across multiple domains, as it is expected to be applied on a diverse set of documents: historical texts, news articles, patent applications, webpages etc. Therefore, we have considered three datasets: CoNLL03 shared task data, MUC7 data and a set of Webpages we have annotated manually. In the experiments throughout the paper, we test the ability of the tagger to adapt to new test domains. Throughout this work, we train on the CoNLL03 data and test on the other datasets *without retraining*. The differences in annotation schemes across datasets created evaluation challenges. We discuss the datasets and the evaluation methods below.

The CoNLL03 shared task data is a subset of Reuters 1996 news corpus annotated with 4 entity types: *PER, ORG, LOC, MISC*. It is important to notice that *both* the training and the development datasets are news feeds from *August* 1996, while the test set contains news feeds from *December* 1996. The named entities mentioned in the test dataset are considerably different from those that appear in the training or the development set. As a result, the test dataset is considerably harder than the development set. **Evaluation:** Following the convention, we report phrase-level F_1 score.

The MUC7 dataset is a subset of the North American News Text Corpora annotated with a wide variety of entities including people, locations, organizations, temporal events, monetary units, and so on. Since there was no direct mapping from temporal events, monetary units, and other entities from MUC7 and the MISC label in the CoNLL03 dataset, we measure performance only on *PER,ORG* and *LOC*. We note that the topic of MUC7 is aviation, with training set talking about plane accidents, the test (formalrun) set talking about rocket launches and the development set (dryrun) talking about balloon launches. **Evaluation:** There are several sources of inconsistency in annotation between MUC7 and CoNLL03. For example, since the MUC7 dataset does not contain the *MISC* label, in the sentence *"balloon, called the Virgin Global Challenger"* , the expression *Virgin Global Challenger* should be labeled as *MISC* according to CoNLL03 guidelines. However, the gold annotation in MUC7 is *"balloon, called the* [*ORG Virgin*] *Global Challenger"*. These and other annotation inconsistencies have prompted us to relax the requirements of finding the exact phrase boundaries and measure performance using token-level F_1.

Webpages - we have assembled and manually annotated a collection of 20 webpages, including per-

Algorithm	Baseline system F_1	Baseline system sec	All Features F_1	All Features sec
Greedy	83.65	11.46	90.57	67.20
Beam size=10	83.68	134.11	90.67	621.80
Beam size=50	83.70	680.21	90.67	2819.94
Viterbi	83.71	1433.73	N/A	N/A

Table 4.1: Phrase-level F_1 performance on CoNLL03 test data vs. Inference time (sec). The results are reported on a quad-core IntelXeon 2.83GHz machine with 8GB of RAM.

sonal, academic and computer-science conference homepages. The dataset contains 783 entities (96-loc, 223-org, 276-per, 188-misc). **Evaluation:** The named entities in the webpages were highly ambiguous and very different from the named entities seen in the training data. For example, the data included sentences such as : *"Hear, O Israel, the Lord our God, the Lord is one."* We could not agree on whether *"O Israel"* should be labeled as *ORG, LOC,* or *PER*. Similarly, we could not agree on whether *"God"* and *"Lord"* is an *ORG* or *PER*. These issues led us to report token-level entity-identification F_1 score for this dataset. That is, if a named entity token was identified as such, we counted it as a correct prediction ignoring the named entity type.

4.2.2 Inference & Chunk Representation

In this section we compare the performance of several inference (decoding) algorithms: greedy left-to-right decoding, Viterbi and beamsearch. It may appear that beamsearch or Viterbi will perform much better than naive greedy left-to-right decoding, which can be seen as beamsearch of size one. The Viterbi algorithm has the limitation that it does not allow incorporating some of the non-local features which will be discussed later, therefore, we cannot use it in our end system. However, it has the appealing quality of finding the most likely assignment to a second-order model, and since the baseline features only have second order dependencies, we have tested it on the baseline configuration.

Table 4.1 compares between the greedy decoding, beamsearch with varying beam size, and Viterbi, both for the system with baseline features and for the end system (to be presented later). Surprisingly, the greedy policy performs well, this phenmenon was also observed in the POS tagging task [143; 121]. The implications are subtle. First, due to the second-order of the model, the greedy decoding is over 100 times faster than Viterbi. The reason is that with the BILOU encoding of four NE types, each token can take 21 states (*O, B-PER, I-PER , U-PER, etc.*). To tag a token, the greedy policy requires 21 comparisons, while the Viterbi requires 21^3, and this analysis carries over to the number of classifier invocations. Furthermore, both beamsearch and Viterbi require transforming the predictions of the classifiers to probabilities as discussed

Rep. Scheme	CoNLL03 Test	CoNLL03 Dev	MUC7 Dev	MUC7 Test
BIO	89.15	**93.61**	86.76	85.15
BILOU	**90.57**	93.28	**88.09**	**85.62**

Table 4.2: End system performance with BILOU and BIO schemes. BILOU outperforms the more widely used BIO.

in [100], incurring additional time overhead. Second, this result reinforces the intuition that global inference over the second-order HMM features does not capture the non-local properties of the task. The reason is that the NEs tend to be short chunks separated by multiple "outside" tokens. This separation "breaks" the Viterbi decision process to independent maximization of assignment over short chunks, where the greedy policy performs well. On the other hand, dependencies between isolated named entity chunks have *longer*-range dependencies and are not captured by second-order transition features, therefore requiring separate mechanisms, which we discuss in Section 4.2.3.

4.2.3 Non-Local Features

The key intuition behind non-local features in NER has been that identical tokens should have identical label assignments. The sample text discussed in the introduction shows one such example, where all occurrences of *"blinker"* are assigned the *PER* label. However, in general, this is not always the case; for example we might see in the same document the word sequences *"Australia"* and *"The bank of Australia"*. The first instance should be labeled as *LOC*, and the second as *ORG*. We consider three approaches proposed in the literature in the following sections. Before continuing the discussion, we note that we found that adjacent documents in the CoNLL03 and the MUC7 datasets often discuss the same entities. Therefore, we ignore document boundaries and analyze global dependencies in 200 and 1000 token windows. These constants were selected by hand after trying a small number of values. We believe that this approach will also make our system more robust in cases when the document boundaries are not given.

Context aggregation

[23] used features that aggregate, for each document, the context tokens appear in. Sample features are: *the longest capitilized sequence of words in the document which contains the current token* and *the token appears before a company marker such as ltd. elsewhere in text*. In this work, we call this type of features *context aggregation features*. Manually designed context aggregation features clearly have low coverage, therefore we used the following approach. Recall that for each token instance x_i, we use as features the tokens in the window

of size two around it: $c_i = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$. When the same token type t appears in several locations in the text, say $x_{i_1}, x_{i_2}, \ldots, x_{i_N}$, for each instance x_{i_j}, in addition to the context features c_{i_j}, we also aggregate the context across all instances within 200 tokens: $C = \cup_{j=1}^{j=N} c_{i_j}$.

Two-stage prediction aggregation

Context aggregation as done above can lead to excessive number of features. [73] used the intuition that some instances of a token appear in easily-identifiable contexts. Therefore they apply a baseline NER system, and use the resulting predictions as features in a second level of inference. We call the technique *two-stage prediction aggregation*. We implemented the token-majority and the entity-majority features discussed in [73]; however, instead of document and corpus majority tags, we used relative frequency of the tags in a 1000 token window.

Extended prediction history

Both context aggregation and two-stage prediction aggregation treat all tokens in the text similarly. However, we observed that the named entities in the beginning of the documents tended to be more easily identifiable and matched gazetteers more often. This is due to the fact that when a named entity is introduced for the first time in text, a canonical name is used, while in the following discussion abbreviated mentions, pronouns, and other references are used. To break the symmetry, when using beamsearch or greedy left-to-right decoding, we use the fact that when we are making a prediction for token instance x_i, we have already made predictions $y_1, \ldots, y_{i-1}$ for token instances $x_1, \ldots, x_{i-1}$. When making the prediction for token instance x_i, we record the label assignment distribution for all token instances for the same token type in the previous 1000 words. That is, if the token instance is *"Australia"*, and in the previous 1000 tokens, the token type *"Australia"* was twice assigned the label *L-ORG* and three times the label *U-LOC*, then the prediction history feature will be: $(L - ORG : \frac{2}{5}; U - LOC : \frac{3}{5})$.

Utility of non-local features

Table 4.3 summarizes the results. Surprisingly, no single technique outperformed the others on all datasets. The extended prediction history method was the best on CoNLL03 data and MUC7 test set. Context aggregation was the best method for MUC7 development set and two-stage prediction was the best for Webpages. Non-local features proved less effective for MUC7 test set and the Webpages. Since the named entities in Webpages have less context, this result is expected for the Webpages. However, we are unsure why MUC7 test set benefits from nonlocal features much less than MUC7 development set. Our key con-

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + Context Aggregation	85.40	89.99	**79.16**	71.53	70.76
3)	(1) + Extended Prediction History	**85.57**	**90.97**	78.56	**74.27**	72.19
4)	(1)+ Two-stage Prediction Aggregation	85.01	89.97	75.48	72.16	**72.72**
5)	All Non-local Features (1-4)	86.53	90.69	81.41	73.61	71.21

Table 4.3: The utility of non-local features. The system was trained on CoNLL03 data and tested on CoNNL03, MUC7 and Webpages. No single technique outperformed the rest on all domains. The combination of all techniques is the most robust.

clusion is that no single approach is better than the rest and that the approaches are complimentary- their combination is the most stable and best performing.

4.2.4 External Knowledge

As we have illustrated in the introduction, NER is a knowledge-intensive task. In this section, we discuss two important knowledge resources– gazetteers and unlabeled text.

Unlabeled Text

Recent successful semi-supervised systems [1; 135] have illustrated that unlabeled text can be used to improve the performance of NER systems. In this work, we analyze a simple technique of using word clusters generated from unlabeled text, which has been shown to improve performance of dependency parsing [72], Chinese word segmentation [80] and NER [91]. The technique is based on word class models, pioneered by [14], which hierarchically clusters words, producing a binary tree as in Figure 4.3.

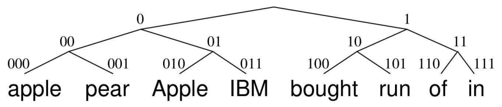


Figure 4.3: An extract from word cluster hierarchy.

The approach is related, but not identical, to distributional similarity (for details, see [14] and [80]). For example, since the words *Friday* and *Tuesday* appear in similar contexts, the Brown algorithm will assign them to the same cluster. Successful abstraction of both as a *day of the week*, addresses the data sparsity problem common in NLP tasks. In this work, we use the implementation and the clusters obtained in [80] from running the algorithm on the Reuters 1996 dataset, a superset of the CoNLL03 NER dataset. Within the binary tree produced by the algorithm, each word can be uniquely identified by its path from the root, and this path can be compactly represented with a bit string. Paths of different depths along the path from

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + External Knowledge	88.55	92.49	84.50	83.23	74.44
3)	(1) + Non-local	86.53	90.69	81.41	73.61	71.21
4)	**All Features**	**90.57**	**93.50**	**89.19**	**86.15**	**74.53**
5)	All Features (train with dev)	90.80	N/A	89.19	86.15	74.33

Table 4.4: End system performance by component. Results confirm that NER is a knowledge-intensive task.

the root to the word provide different levels of word abstraction. For example, paths at depth 4 closely correspond to POS tags. Since word class models use large amounts of unlabeled data, they are essentially a semi-supervised technique, which we use to considerably improve the performance of our system.

In this work, we used path prefixes of length 4,6,10, and 20. When Brown clusters are used as features in the following sections, it implies that all features in the system which contain a word form will be duplicated and a new set of features containing the paths of varying length will be introduced. For example, if the system contains the feature *concatenation of the current token and the system prediction on the previous word*, four new features will be introduced which are concatenations of the previous prediction and the 4,6,10,20 length path-representations of the current word.

Gazetteers

An important question at the inception of the NER task was whether machine learning techniques are necessary at all, and whether simple dictionary lookup would be sufficient for good performance. Indeed, the baseline for the CoNLL03 shared task was essentially a dictionary lookup of the entities which appeared in the training data, and it achieves 71.91 F_1 score on the test set [140]. It turns out that while problems of coverage and ambiguity prevent straightforward lookup, injection of gazetteer matches as features in machine-learning based approaches is critical for good performance [24; 70; 141; 47]. Given these findings, several approaches have been proposed to automatically extract comprehensive gazetteers from the web and from large collections of unlabeled text [41; 114] with limited impact on NER. Recently, [141; 70] have successfully constructed high quality and high coverage gazetteers from Wikipedia.

It is important to note that gazetteers have a property which is often overlooked in the NLP community. It is well known that words are often ambiguous. For example, the word *"Bill"* can refer to a person's name or a piece of legislation. The vast majority of work on word representations such as the Brown Clusters covered in Chapter 4.2.4 and word embeddings covered in Chapter 5 are context-independent, and are unable to capture the context-specific meaning of ambiguous words. At best, we can hope that the represetation induced for such words would be a mixture of senses, but at worst, it can be a representation of the most common sense, which will lead to outlier features for minority senses. One possibility is to develop context-sensitive embeddings, such as those used by [62]. Unfortunately, context-sensitive embeddings were not proved to be effective on multiple tasks, and we were unable to improve the performance of our baseline NER system with our implementation of techniques discussed in [62]. Instead we notice that multi-word phrases are much less ambiguous than words because the words in a phrase provide contexts for one another. For example *"Bill Clinton"* clearly refers to a person, while *"Bill of Rights 1689"* clearly refers to legislation.

In this work, we use a collection of 14 high-precision, low-recall lists extracted from the web that cover common names, countries, monetary units, temporal expressions, etc. While these gazetteers have excellent accuracy, they do not provide sufficient coverage. We note that [81] have used distributed k-means clustering of 20M unique queries from an anonymized query log in a 700 billion token web corpus and show significant performance improvement on the NER task. However, we follow [141; 70] and construct 16 high quality and high coverage gazetteers of multi-token expressions from Wikipedia which collectively contain over 1.5M entities. We note that matches against different gazetteers are represented as separate features in the system, which allows us to trust each gazetteer to a different degree. In the rest of this section, we discuss the construction of gazetteers from Wikipedia.

Wikipedia is an open, collaborative encyclopedia with several attractive properties. (1) It is kept updated manually by it collaborators, hence new entities are constantly added to it. (2) Wikipedia contains redirection pages, mapping several variations of spelling of the same name to one canonical entry. For example, *Suker* is redirected to an entry about *Davor Šuker*, the Croatian footballer (3) The entries in Wikipedia are manually tagged with categories. For example, the entry about the *Microsoft* in Wikipedia has the following categories: *Companies listed on NASDAQ; Cloud computing vendors; etc.*

Both [141] and [70] used the free-text description of the Wikipedia entity to reason about the entity type. We use a simpler method to extract high coverage and high quality gazetteers from Wikipedia. By inspection of the CoNLL03 shared task annotation guidelines and of the training set, we manually aggregated several categories into a higher-level concept (not necessarily NER type). When a Wikipedia entry was tagged by one of the categories in the table, it was added to the corresponding gazetteer. The exact mapping from Wikipedia categories to gazetteers is shown in Figure 4.4.

Utility of External Knowledge

Table 4.5 summarizes the results of the techniques for injecting external knowledge. It is important to note that, although the world class model was learned on the superset of CoNLL03 data, and although the

1)**People**: *people, births, deaths*. Extracts 494,699 Wikipedia titles and 382,336 redirect links. 2)**Organizations**: *cooperatives, federations, teams, clubs, departments, organizations, organisations, banks, legislatures, record labels, constructors, manufacturers, ministries, ministers, military units, military formations, universities, radio stations, newspapers, broadcasters, political parties, television networks, companies, businesses, agencies*. Extracts 124,403 titles and 130,588 redirects. 3)**Locations**: *airports, districts, regions, countries, areas, lakes, seas, oceans, towns, villages, parks, bays, bases, cities, landmarks, rivers, valleys, deserts, locations, places, neighborhoods*. Extracts 211,872 titles and 194,049 redirects. 4)**Named Objects**: *aircraft, spacecraft, tanks, rifles, weapons, ships, firearms, automobiles, computers, boats*. Extracts 28,739 titles and 31,389 redirects. 5)**Art Work**: *novels, books, paintings, operas, plays*. Extracts 39,800 titles and 34037 redirects. 6)**Films**: *films, telenovelas, shows, musicals*. Extracts 50,454 titles and 49,252 redirects. 7)**Songs**: *songs, singles, albums*. Extracts 109,645 titles and 67,473 redirects. 8)**Events**: *playoffs, championships, races, competitions, battles*. Extracts 20,176 titles and 15,182 redirects.

Figure 4.4: Mapping of Wikipedia categories to gazetteers.

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + Gazetteer Match	87.22	91.61	85.83	80.43	74.46
3)	(1) + Word Class Model	86.82	90.85	80.25	79.88	72.26
4)	All External Knowledge	88.55	92.49	84.50	83.23	74.44

Table 4.5: Utility of external knowledge. The system was trained on CoNLL03 data and tested on CoNNL03, MUC7 and Webpages.

Wikipedia gazetteers were constructed based on CoNLL03 annotation guidelines, these features proved extremely good on all datasets. Word class models discussed in Section 4.2.4 are computed offline, are available online[2], and provide an alternative to traditional semi-supervised learning. It is important to note that the word class models and the gazetteers and independednt and accumulative. Furthermore, despite the number and the gigantic size of the extracted gazetteers, the gazeteers alone are not sufficient for adequate performance. When we modified the CoNLL03 baseline to include gazetteer matches, the performance went up from 71.91 to 82.3 on the CoNLL03 test set, below our baseline system's result of 83.65. When we have injected the gazetteers into our system, the performance went up to 87.22. Word class model and nonlocal features further improve the performance to 90.57 (see Table 4.4), by more than 3 F_1 points.

4.2.5 Final System Performance Analysis

As a final experiment, we have trained our system both on the training and on the development set, which gave us our best F_1 score of 90.8 on the CoNLL03 data, yet it failed to improve the performance on other datasets. Table 4.4 summarizes the performance of the system.

[2]http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz

Dataset	Stanford-NER	LBJ-NER
MUC7 Test	80.62	85.71
MUC7 Dev	84.67	87.99
Webpages	72.50	74.89
Reuters2003 test	87.04	90.74
Reuters2003 dev	92.36	93.94

Table 4.6: Comparison: token-based F_1 score of LBJ-NER and Stanford NER tagger across several domains

Next, we have compared the performance of our system to that of the Stanford NER tagger, across the datasets discussed above. We have chosen to compare against the Stanford tagger because to the best of our knowledge, it is the best publicly available system which is trained on the same data. We have downloaded the Stanford NER tagger and used the strongest provided model trained on the CoNLL03 data with distributional similarity features. The results we obtained on the CoNLL03 test set were consistent with what was reported in [45]. Our goal was to compare the performance of the taggers across several datasets. For the most realistic comparison, we have presented each system with a raw text, and relied on the system's sentence splitter and tokenizer. When evaluating the systems, we matched against the gold tokenization ignoring punctuation marks. Table 4.6 summarizes the results. Note that due to differences in sentence splitting, tokenization and evaluation, these results are not identical to those reported in Table 4.4. Also note that in this experiment we have used token-level accuracy on the CoNLL dataset as well. Finally, to complete the comparison to other systems, in Table 4.7 we summarize the best results reported for the CoNLL03 dataset in literature.

4.2.6 Conclusions

We have presented a simple model for NER that uses expressive features to achieve new state of the art performance on the Named Entity recognition task. We explored four fundamental design decisions: text chunks representation, inference algorithm, using non-local features and external knowledge. We showed that BILOU encoding scheme significantly outperforms BIO and that, surprisingly, a conditional model that does not take into account interactions at the output level performs comparably to beamsearch or Viterbi, while being considerably more efficient computationally. We analyzed several approaches for modeling non-local dependencies and found that none of them clearly outperforms the others across several datasets. Our experiments corroborate recently published results indicating that word class models learned on unlabeled text can be an alternative to the traditional semi-supervised learning paradigm. NER proves to be a knowledge-intensive task, and it was reassuring to observe that knowledge-driven techniques adapt well across several domains. We observed consistent performance gains across several domains, most interest-

	System	Resources Used	F_1
+	LBJ-NER	Wikipedia, Nonlocal Features, Word-class Model	90.80
-	[135]	Semi-supervised on 1G-word unlabeled data	89.92
-	[1]	Semi-supervised on 27M-word unlabeled data	89.31
-	[70]	Wikipedia	88.02
-	[73]	Non-local Features	87.24
-	[69]	Non-local Features	87.17
+	[45]	Non-local Features	86.86

Table 4.7: Results for CoNLL03 data reported in the literature. publicly available systems marked by +.

ingly in Webpages, where the named entities had less context and were different in nature from the named entities in the training set. Our system significantly outperforms the current state of the art and is available to download under a research license.

Chapter 5

A Study of Word Representations

Recently published results [91; 80; 72; 112; 62] show that word representations learned from unlabeled text can significantly improve the performance of an NLP system and can act as an alternative to the traditional semi-supervised learning paradigm. While dedicated semi-supervised approaches, such as [1; 135; 11] exist and can be quite successful, they require task-specific and model-specific math derivations followed by corresponding re-design of the (supervised training) code. The appealing property of word representations is that once computed, they provide a general approach for capturing similarity between words (or abstracting over words). This allows them to be potentially used in multiple applications in a straightforward and easy way to address the data sparsity problem common in NLP [68; 64]. Additionally, word representations can be viewed as a way to encode knowledge extracted from large quantities of unlabeled text. As we have showed in Chapter 4, NER is a knowledge-intensive task, and throughout this thesis we argue that so is NLP in general. Therefore, a general infrastructure for capturing word similarity, which can be easily injected into an arbitrary NLP application seems extremely valuable. With several such approaches proposed, it is natural to ask two questions:

1. Which approach is most useful for the current state-of-the-art NLP systems?
2. Are these approaches general enough to be useful in multiple NLP applications?

In this chapter we present an empirical study comparing between the different word representations on two NLP tasks: NER and shallow parsing (chunking).

5.1 Types of Word Representations

In this section we cover the existing approaches for representing words. We note that in essence, all the approaches described below share the "distributional hypothesis" assumption [58; 46], which states that words appearing in similar contexts share similar meanings. Hence, in a sense, the differences between the approaches are purely technical - how to capture the distributional properties of the words. Nevertheless, these differences are crucial, for example, some approaches may attempt to model the polysemy of words, while others may not. Some approaches represent a word by clustering it with similar words and assigning each cluster a unique identifier. Yet other approaches attempt to induce latent properties of the words and learn them from their context.

5.1.1 Distributional representations

Distributional word representations are based upon a co-occurrence matrix F of size $W \times C$, where W is the vocabulary size, each row F_w is the initial representation of word w, and each column F_c is some context. [125] and [147] describe a handful of possible design decisions in constructing F, including choice of context types (left window? right window? size of window?) and type of frequency count (raw? binary? tf-idf?). F_w has dimensionality W, which can be too large to use F_w as features for word w in a supervised model. One can map F to matrix f of size $W \times d$, where $d \ll C$, using some function g, where $f = g(F)$. f_w represents word w as a vector with d dimensions. The choice of g is another important design decision, although perhaps not as important as the statistics used to initially construct F.

The self-organizing semantic map [116] is a distributional technique that maps words to two dimensions, such that syntactically and semantically related words are nearby [61; 60].

LSA [39; 76], LSI, and LDA [12] induce distributional representations over F in which each column is a *document* context. (In most of the other approaches discussed, the columns represent word contexts.) In LSA, g computes the SVD of F.

Hyperspace Analogue to Language (HAL) is another early distributional approach [82; 83] to inducing word representations. They compute F over a corpus of 160 million word tokens with a vocabulary size W of 70K word types. There are $2 \cdot W$ types of context (columns): The first or second W are counted if the word c occurs within a window of 10 to the left or right of the word w, respectively. f is chosen by taking the 200 columns (out of 140K in F) with the highest variances.

ICA is another technique to transform F into f [153; 154; 155]. ICA is expensive, and the largest vocabulary size used in these works was only 10K. As far as we know, ICA methods have not been used when the size of the vocabulary W is 100K or more.

Explicitly computing the initial cooccurrence matrix F can be memory-intensive, and transforming F to f can be time-consuming. It is preferable that F never be computed explicitly, and that f be constructed *incrementally*. [159] describe an incremental approach to inducing LSA and LDA topic models over 270 millions word tokens with a vocabulary of 315K word types. This is similar in magnitude to our experiments.

Another incremental approach to constructing f is using a random projection: Linear mapping g is multiplying F by a random matrix chosen *a priori*. This *random indexing* method is motivated by the Johnson-Lindenstrauss lemma, which states that for certain choices of random matrix, if d is sufficiently large, then the original distances between words in F will be preserved in f [124]. [67] uses this technique to produce 100-dimensional representations of documents. [123] was the first author to use random indexing using narrow context. [125] does a battery of experiments exploring different design decisions involved in constructing F, prior to using random indexing. However, like all the works cited above, [125] only uses distributional representation to improve existing systems for one-shot classification tasks, such as IR, WSD, semantic knowledge tests, and text categorization. It is not well-understood what settings are appropriate to induce distributional word representations for structured prediction tasks (like parsing and MT) and sequence labeling tasks (like chunking and NER). Previous research has achieved repeated successes on these tasks using clustering representations (Section 5.1.2) and distributed representations (Section 5.1.3), so we focus on these representations in our work.

5.1.2 Clustering-based word representations

Another type of word representation is to induce a clustering over words. Clustering methods and distributional methods can overlap. For example, [103] begin with a co-occurrence matrix and transform this matrix into a clustering.

Brown clustering

The Brown algorithm is a hierarchical clustering algorithm which clusters words to maximize the mutual information of bigrams [14]. So it is a class-based bigram language model. It runs in time $O(V \cdot K^2)$, where V is the size of the vocabulary and K is the number of clusters.

The hierarchical nature of the clustering means that we can choose the word class at several levels in the hierarchy, which can compensate for poor clusters of a small number of words. One downside of Brown clustering is that it is based solely on bigram statistics, and does not consider word usage in a wider context.

Brown clusters have been used successfully in a variety of NLP applications: NER [91; 80; 112], PCFG parsing [16], dependency parsing [72; 135], and semantic dependency parsing [164].

[86] presents algorithms for inducing hierarchical clusterings based upon word bigram *and trigram* statistics. [148] presents an extension to the Brown clustering algorithm, and learn hierarchical clusterings of words *as well as phrases*, which they apply to POS tagging.

Other work on cluster-based word representations

[81] present a K-means-like non-hierarchical clustering algorithm for phrases, which uses MapReduce. HMMs can be used to induce a soft clustering, specifically a multinomial distribution over possible clusters (hidden states). [79] use an HMM-LDA model to improve POS tagging and Chinese Word Segmentation. [62] induce a fully-connected HMM, which emits a multinomial distribution over possible vocabulary words. They perform hard clustering using the Viterbi algorithm. (Alternately, they could keep the soft clustering, with the representation for a particular word token being the posterior probability distribution over the states.) However, the CRF chunker in [62], which uses their HMM word clusters as extra features, achieves F1 lower than a baseline CRF chunker [128]. [53] use an HMM to assign POS tags to words, which in turns improves the accuracy of the PCFG-based Hebrew parser. [36] use a latent-variable language model to improve semantic role labeling.

5.1.3 Distributed representations

Another approach to word representation is to learn a distributed representation. (Not to be confused with *distributional* representations.) A distributed representation is dense, low-dimensional, and real-valued. Distributed word representations are called *word embeddings*. Each dimension of the embedding represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties. A distributed representation is compact, in the sense that it can represent an exponential number of clusters in the number of dimensions.

Word embeddings are typically induced using *neural language models*, which use neural networks as the underlying predictive model [5]. Historically, training and testing of neural language models has been slow, scaling as the size of the vocabulary for each model computation [6; 8; 9]. However, many approaches have been proposed in recent years to eliminate that linear dependency on vocabulary size [96; 28; 95] and allow scaling to very large training corpora.

[28] presented a neural language model that could be trained over billions of words, because the gradient of the loss was computed stochastically over a small sample of possible outputs, in a spirit similar to [9]. This neural model of [28] was refined and presented in greater depth in [7].

The model is discriminative and non-probabilistic. For each training update, we read an n-gram $x = (w_1, \ldots, w_n)$ from the corpus. The model concatenates the learned embeddings of the n words, giving $e(w_1) \oplus \cdots \oplus e(w_n)$, where e is the lookup table and $\oplus$ is concatenation. We also create a *corrupted* or *noise* n-gram $\tilde{x} = (w_1, \ldots, w_{n-q}, \tilde{w_n})$, where $\tilde{w_n} \neq w_n$ is chosen uniformly from the vocabulary.[1] For

[1] In [28], the middle word in the n-gram is corrupted. In [7], the last word in the n-gram is corrupted.

convenience, we write $e(x)$ to mean $e(w_1) \oplus \cdots \oplus e(w_n)$. We predict a score $s(x)$ for x by passing $e(x)$ through a single hidden layer neural network. The training criterion is that n-grams that are present in the training corpus like x must have a score at least some margin higher than corrupted n-grams like $\tilde{x}$. Specifically: $L(x) = \max(0, 1 - s(x) + s(\tilde{x}))$. We minimize this loss stochastically over the n-grams in the corpus, doing gradient descent simultaneously over the neural network parameters *and* the embedding lookup table.

We implemented the approach of [28], with the following differences:

• We did not achieve as low log-ranks on the English Wikipedia as the authors reported in [7], despite initially attempting to have identical experimental conditions.

• We corrupt the *last* word of each n-gram.

• We had a separate learning rate for the embeddings and for the neural network weights. We found that the embeddings should have a learning rate generally 1000–32000 times higher than the neural network weights. Otherwise, the unsupervised training criterion drops slowly.

• Although their sampling technique makes training fast, testing is still expensive when the size of the vocabulary is large. Instead of cross-validating using the log-rank over the validation data as they do, we instead used the moving average of the training loss on training examples before the weight update.

HLBL embeddings

The log-bilinear model [94] is a probabilistic and linear neural model. Given an n-gram, the model concatenates the embeddings of the $n-1$ first words, and learns a linear model to predict the embedding of the last word. The similarity between the predicted embedding and the current actual embedding is transformed into a probability by exponentiating and then normalizing. [95] speed up model evaluation during training and testing by using a hierarchy to exponentially filter down the number of computations that are performed. This hierarchical evaluation technique was first proposed by [96]. The model, combined with this optimization, is called the *hierarchical log-bilinear (HLBL)* model.

5.2 Supervised evaluation tasks

We evaluate the hypothesis that one can take an existing, near state-of-the-art, supervised NLP system, and improve its accuracy by including word representations as word features. This technique for turning a supervised approach into a semi-supervised one is general and task-agnostic.

However, we wish to find out if certain word representations are preferable for certain tasks. [81] finds

- Word features: w_i for i in $\{-2, -1, 0, +1, +2\}$, $w_i \wedge w_{i+1}$ for i in $\{-1, 0\}$.
- Tag features: w_i for i in $\{-2, -1, 0, +1, +2\}$, $t_i \wedge t_{i+1}$ for i in $\{-2, -1, 0, +1\}$. $t_i \wedge t_{i+1} \wedge t_{i+2}$ for i in $\{-2, -1, 0\}$.
- Embedding features [if applicable]: $e_i[d]$ for i in $\{-2, -1, 0, +1, +2\}$, where d ranges over the dimensions of the embedding e_i.
- Brown features [if applicable]: $substr(b_i, 0, p)$ for i in $\{-2, -1, 0, +1, +2\}$, where $substr$ takes the p-length prefix of the Brown cluster b_i.

Table 5.1: Features templates used in the CRF chunker.

that the representations that are good for NER are poor for search query classification, and vice-versa. We apply clustering and distributed representations to NER and chunking, which allows us to compare our semi-supervised models to those of [1] and [135].

5.2.1 Chunking

Chunking is a syntactic sequence labeling task. We follow the conditions in the CoNLL-2000 shared task [126].

The linear CRF chunker of [128] is a standard near-state-of-the-art baseline chunker. In fact, many off-the-shelf CRF implementations now replicate [128], including their choice of feature set:

- CRF++ by Taku Kudo (`http://crfpp.sourceforge.net/`)
- crfsgd by Léon Bottou (`http://leon.bottou.org/projects/sgd`)
- CRFsuite by by Naoaki Okazaki (`http://www.chokkan.org/software/crfsuite/`)

We use CRFsuite because it makes it simple to modify the feature generation code, so one can easily add new features. We use SGD optimization, and enable negative state features and negative transition features. (*"feature.possible_transitions=1, feature.possible_states=1"*)

Table 5.1 shows the features in the baseline chunker. As you can see, the Brown and embedding features are unigram features, and do not participate in conjunctions like the word features and tag features do. [72] sees further accuracy improvements on dependency parsing when using word representations in compound features.

The data comes from the Penn Treebank, and is newswire from the Wall Street Journal in 1989. Of the 8936 training sentences, we used 1000 randomly sampled sentences (23615 words) for development. We trained models on the 7936 training partition sentences, and evaluated their F1 on the development set. After choosing hyperparameters to maximize the dev F1, we would retrain the model using these hyperparameters on the full 8936 sentence training set, and evaluate on test. One hyperparameter was l2-regularization sigma, which for most models was optimal at 2 or 3.2. The word embeddings also required a scaling hyperparameter, as described in Section 5.4.2.

5.2.2 Named entity recognition

NER is typically treated as a sequence prediction problem. Following [112], we use the regularized averaged perceptron model. [112] describe different sequence encoding like BILOU and BIO, and show that the BILOU encoding outperforms BIO, and the greedy inference performs competitively to Viterbi while being significantly faster. Accordingly, we use greedy inference and BILOU text chunk representation. We use the publicly available implementation from [112][2]. In our baseline experiments, we remove gazetteers and non-local features [73]. However, we also run experiments that include these features, to understand if the information they provide mostly overlaps with that of the word representations.

After each epoch over the training set, we measured the accuracy of the model on the development set. Training was stopped after the accuracy on the development set did not improve for 10 epochs, generally about 50–80 epochs total. The epoch that performed best on the development set was chosen as the final model. We report phrase-based F1 performance on all datasets[3].

We use the following baseline set of features from [163]:

- Previous two predictions y_{i-1} and y_{i-2}
- Current word x_i
- x_i word type information: all-capitalized, is-capitalized, all-digits, alphanumeric, etc.
- Prefixes and suffixes of x_i, if the word contains hyphens, then the tokens between the hyphens
- Tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
- Capitalization pattern in the window c
- Conjunction of c and y_{i-1}.

Word representation features, if present, are used the same way as in Table 5.1[4].

Unlike in our chunking experiments, after we chose the best model on the development set, we used that model on the test set too. (In chunking, after finding the best hyperparameters on the development set, we would combine the dev and training set and training a model over this combined set, and then evaluate

[2]`http://cogcomp.cs.illinois.edu/page/software`

[3]These settings are somewhat different than the ones used in Chapter 4, hence, the results reported in this chapter are somewhat different as well.

[4]In Chapter 4, we have generated conjunctions of words and Brown clusters. However, conjunctions of words with the embedded representations proved to be very slow. Hence for systematic comparison of Brown clusters and the embedded representations we have removed these conjunctions altogether. This is another source of inconsistency between the results reported in this chapter and the results reported in Chapter 4.

on test.)

The standard evaluation benchmark for NER is the CoNLL03 shared task dataset drawn from the Reuters newswire. The training set contains 204K words (14K sentences, 946 documents), the test set contains 46K words (3.5K sentences, 231 documents), and the development set contains 51K words (3.3K sentences, 216 documents).

We also evaluated on an out-of-domain (OOD) dataset, the MUC7 formal run (59K words). MUC7 has a different annotation standard than the CoNLL03 data. It has several NE types that don't appear in CoNLL03: money, dates, and numeric quantities. CoNLL03 has MISC, which is not present in MUC7. To evaluate on MUC7, we perform the following postprocessing steps prior to evaluation:

1. In the gold-standard MUC7 data, discard (label as 'O') all NEs with type NUMBER/MONEY/DATE.
2. In the predicted model output on MUC7 data, discard (label as 'O') all NEs with type MISC.

These postprocessing steps will adversely affect all NER models across-the-board, nonetheless allowing us to compare different models in a controlled manner.

5.3 Unlabeled Data

Unlabeled data is used for inducing the word representations. We used the RCV1 corpus, which contains one year of Reuters English newswire, from August 1996 to August 1997, about 63 millions words in 3.3 million sentences[5]. We left case intact in the corpus. By comparison, [28] downcases words and delexicalizes numbers.

We use a preprocessing technique proposed by [80, p. 51], which was later used by [72]: Remove all sentences that are less than 90% lowercase a–z. We assume that whitespace is not counted, although this is not specified in Liang's thesis. We call this preprocessing step *cleaning*.

In [146], we found that all word representations performed better on the supervised task when they were induced on the clean unlabeled data, both embeddings and Brown clusters. This is the case even though the cleaning process was very aggressive, and discarded more than half of the sentences. According to the evidence and arguments presented in [7], the non-convex optimization process for [28] embeddings might be adversely affected by noise and the statistical sparsity issues regarding rare words, especially at the beginning of training. For this reason, we hypothesize that learning representations over the most

[5]In Chapter 4, the Brown Clusters were induced on the BLLIP corpus, which we believe is a subset of the Wall Street Journal rather than on RCV1, as done here. We found the RCV1-induced word representations significantly more useful for NER and marginally more useful for chunking than word representations induced on BLLIP corpus. However these representations can be combined and indeed we observed performance improvement when using in parallel word representations induced from several resources (RCV1, Wikipedia, BLLIP) as different feature groups. We omit these resutls from this work.

frequent words first and gradually increasing the vocabulary—a *curriculum* training strategy [40; 7; 132]—would provide better results than cleaning.

After cleaning, there are 37 million words (58% of the original) in 1.3 million sentences (41% of the original). The cleaned RCV1 corpus has 269K word types. This is the vocabulary size, i.e. how many word representations were induced. Note that cleaning is applied only to the unlabeled data, not to the labeled data used in the supervised tasks.

RCV1 is a superset of the CoNLL03 corpus. For this reason, NER results that use RCV1 word representations are a form of transductive learning.

5.4 Experiments and Results

5.4.1 Details of inducing word representations

The Brown clusters took roughly 3 days to induce, when we induced 1000 clusters, the baseline in prior work [72; 112]. We also induced 100, 320, and 3200 Brown clusters, for comparison. (Because Brown clustering scales quadratically in the number of clusters, inducing 10000 clusters would have been prohibitive.) Because Brown clusters are hierarchical, we can use cluster supersets as features. We used clusters at path depth 4, 6, 10, and 20 [112]. These are the prefixes used in Table 5.1.

The [28] (C&W) embeddings were induced over the course of a few weeks, and trained for about 50 epochs. One of the difficulties in inducing these embeddings is that there is no stopping criterion defined, and that the quality of the embeddings can keep improving as training continues. Collobert (p.c.) simply leaves one computer training his embeddings indefinitely. We induced embeddings with 25, 50, 100, or 200 dimensions over 5-gram windows. In comparison to [146], we use improved C&W embeddings in this work:

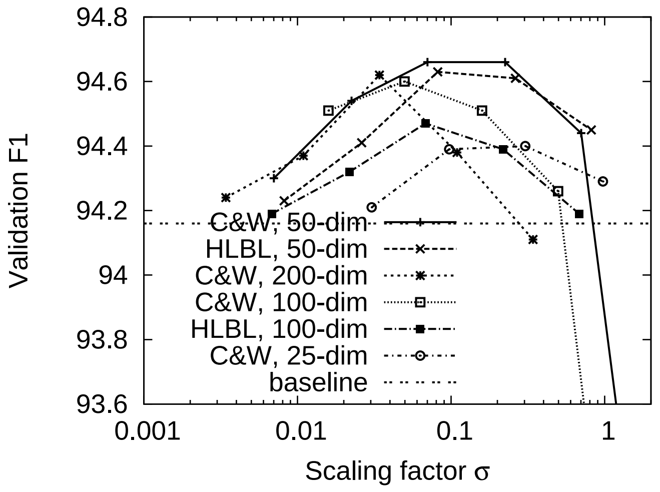
• They were trained for 50 epochs, not just 20 epochs.

• We initialized all embedding dimensions uniformly in the range [-0.01, +0.01], not [-1,+1]. For rare words, which are typically updated only 143 times per epoch[6], and given that our embedding learning rate was typically 1e-6 or 1e-7, this means that rare word embeddings will be concentrated around zero, instead of spread out randomly.

The HLBL embeddings were trained for 100 epochs (7 days).[7] Unlike our [28] embeddings, we did not extensively tune the learning rates for HLBL. We used a learning rate of 1e-3 for both model parameters

[6]A rare word will appear 5 (window size) times per epoch as a positive example, and 37M (training examples per epoch) / 269K (vocabulary size) = 138 times per epoch as a corruption example.

[7]The HLBL model updates require fewer matrix multiplies than [28] model updates. Additionally, HLBL models were trained on a GPGPU, which is faster than conventional CPU arithmetic.

(a) 

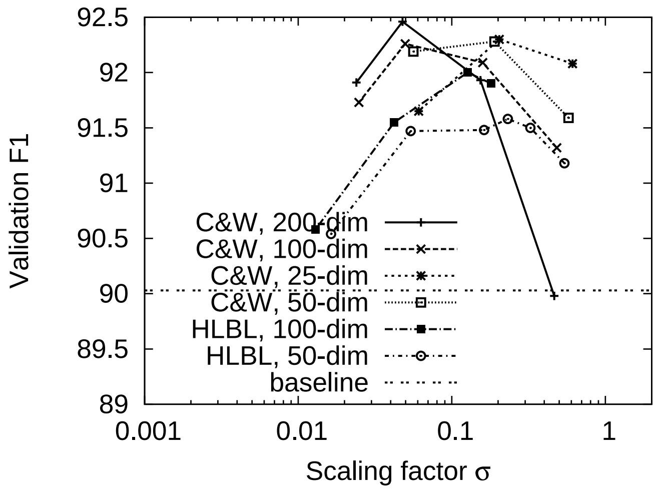
(b) 

Figure 5.1: Effect as we vary the scaling factor σ (Equation 5.1) on the validation set F1. We experiment with [28] and HLBL embeddings of various dimensionality. (a) Chunking results. (b) NER results.

and embedding parameters. We induced embeddings with 100 dimensions over 5-gram windows, and embeddings with 50 dimensions over 5-gram windows. Embeddings were induced over one pass approach using a random tree, not two passes with an updated tree and embeddings re-estimation.

5.4.2 Scaling of Word Embeddings

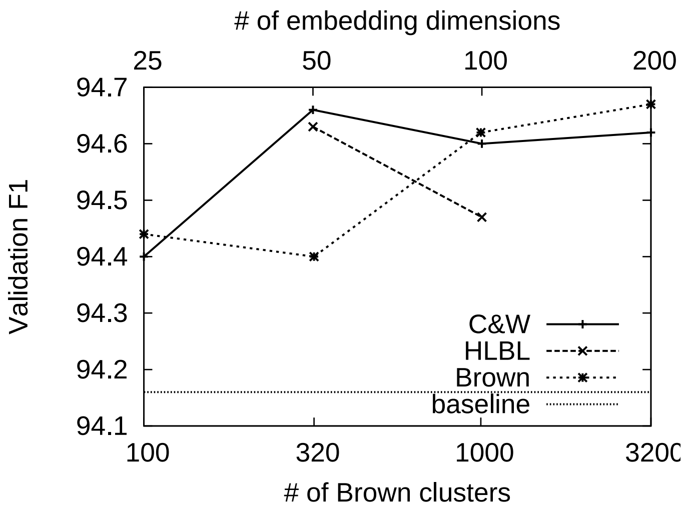
Like many NLP systems, the baseline system contains only binary features. The word embeddings, however, are real numbers that are not necessarily in a bounded range. If the range of the word embeddings is too large, they will exert more influence than the binary features.

We generally found that embeddings had zero mean. We can scale the embeddings by a hyperparameter, to control their standard deviation. Assume that the embeddings are represented by a matrix E:

$$E \leftarrow \sigma \cdot E / stddev(E) \tag{5.1}$$

σ is a scaling constant that sets the new standard deviation after scaling the embeddings.

(a)



(b)

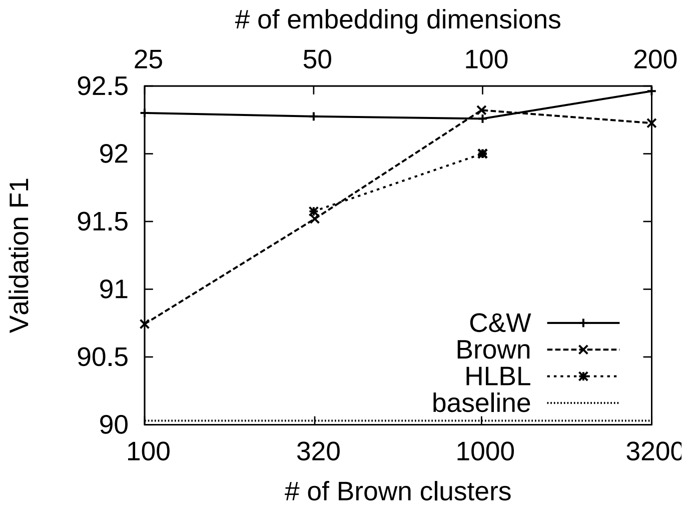


Figure 5.2: Effect as we vary the capacity of the word representations on the validation set F1. (a) Chunking results. (b) NER results.

Figure 5.1 shows the effect of scaling factor σ on both supervised tasks. We were surprised to find that on both tasks, across [28] and HLBL embeddings of various dimensionality, that all curves had similar shapes and optima. This is one contributions of our work. In [146], we were not able to prescribe a default value for scaling the embeddings. However, these curves demonstrate that a reasonable choice of scale factor is such that the embeddings have a standard deviation of 0.1.

5.4.3 Capacity of Word Representations

There are capacity controls for the word representations: number of Brown clusters, and number of dimensions of the word embeddings. Figure 5.2 shows the effect on the validation F1 as we vary the capacity of the word representations.

In general, it appears that more Brown clusters are better. We would like to induce 10000 Brown clusters, however this would take several months.

In [146], we hypothesized on the basis of solely the HLBL NER curve that higher-dimensional word embeddings would give higher accuracy. Figure 5.2 shows that this hypothesis is not true. For NER, the C&W curve is almost flat, and we were suprised to find the even 25-dimensional C&W word embeddings work so well. For chunking, 50-dimensional embeddings had the highest validation F1 for both C&W and HLBL. These curves indicates that the optimal capacity of the word embeddings is task-specific.

System	Dev	Test
Baseline	94.16	93.79
HLBL, 50-dim	94.63	94.00
C&W, 50-dim	94.66	94.10
Brown, 3200 clusters	**94.67**	**94.11**
Brown+HLBL, 37M	94.62	94.13
C&W+HLBL, 37M	94.68	94.25
Brown+C&W+HLBL, 37M	94.72	94.15
Brown+C&W, 37M	94.76	94.35
[1], 15M	-	94.39
[135], 15M	-	94.67
[135], 1B	-	**95.15**

Table 5.2: Final chunking F1 results. In the last section, we show how many unlabeled words were used.

5.4.4 Final results

Table 5.2 shows the final chunking results and Table 5.3 shows the final NER F1 results. We compare to the state-of-the-art methods of [1], [135], and—for NER—[81]. Tables 5.2 and 5.3 show that accuracy can be increased further by combining the features from different types of word representations. But, if only one word representation is to be used, Brown clusters have the highest accuracy. Given the improvements to the C&W embeddings since [146], C&W embeddings outperform the HLBL embeddings. On chunking, there is only a minute difference between Brown clusters and the embeddings. Combining representations leads to small increases in the test F1. In comparison to chunking, combining different word representations on NER seems gives larger improvements on the test F1.

On NER, Brown clusters are superior to the word embeddings. Since much of the NER F1 is derived

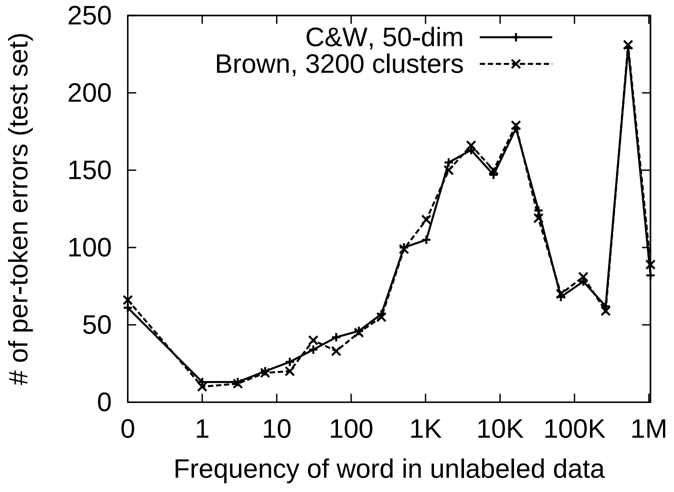
System	Dev	Test	MUC7
Baseline	90.03	84.39	67.48
Baseline+Nonlocal	91.91	86.52	71.80
HLBL 100-dim	92.00	88.13	75.25
Gazetteers	92.09	87.36	77.76
C&W 50-dim	92.27	87.93	75.74
Brown, 1000 clusters	92.32	**88.52**	**78.84**
C&W 200-dim	**92.46**	87.96	75.51
C&W+HLBL	92.52	88.56	78.64
Brown+HLBL	92.56	88.93	77.85
Brown+C&W	92.79	89.31	80.13
HLBL+Gaz	92.91	89.35	79.29
C&W+Gaz	92.98	88.88	81.44
Brown+Gaz	**93.25**	**89.41**	**82.71**
[81], 3.4B	-	88.44	-
[1], 27M	93.15	89.31	-
[135], 37M	93.66	89.36	-
[135], 1B	**94.48**	89.92	-
All (Brown+C&W+HLBL+Gaz), 37M	93.17	90.04	82.50
All+Nonlocal, 37M	93.95	90.36	84.15
[81], 700B	-	**90.90**	-

Table 5.3: Final NER F1 results, showing the cumulative effect of adding word representations, non-local features, and gazetteers to the baseline. To speed up training, in combined experiments (C&W plus another word representation), we used the 50-dimensional C&W embeddings, not the 200-dimensional ones. In the last section, we show how many unlabeled words were used.

from decisions made over rare words, we suspected that Brown clustering has a superior representation for rare words. Brown makes a single hard clustering decision, whereas the embedding for a rare word is close to its initial value since it hasn't received many training updates (see Footnote 6). Figure 5.3 shows the total number of per-token errors incurred on the test set, depending upon the frequency of the word token in the unlabeled data. For NER, Figure 5.3 (b) shows that most errors occur on rare words, and that Brown clusters do indeed incur fewer errors for rare words. This supports our hypothesis that, for rare words, Brown clustering produces better representations than word embeddings that haven't received sufficient training updates. For chunking, Brown clusters and C&W embeddings incur almost identical numbers of errors, and errors are concentrated around the more common words. We hypothesize that *non-rare* words have good representations, regardless of the choice of word representation technique. For tasks like chunking in which a syntactic decision relies upon looking at several token simultaneously, compound features that use the word representations might increase accuracy more [72].

Using word representations in NER brought larger gains on the out-of-domain data than on the in-domain data. We were surprised by this result, because the OOD data was not even used during the unsupervised word representation induction, as was the in-domain data. We are curious to investigate this

(a)



(b)

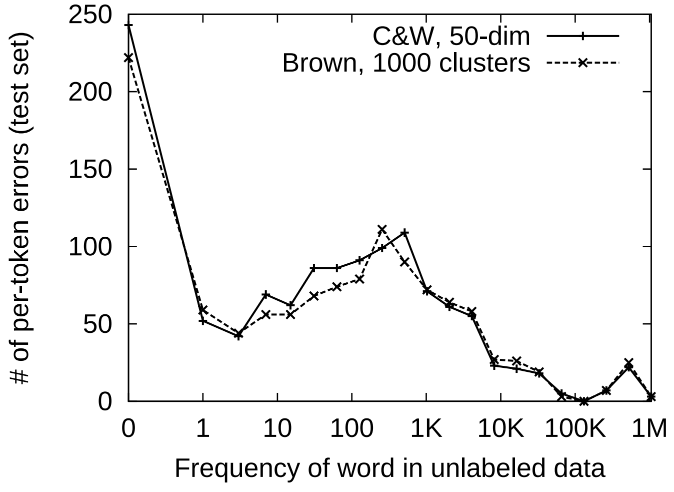


Figure 5.3: For word tokens that have different frequency in the unlabeled data, what is the total number of per-token errors incurred on the test set? (a) Chunking results. (b) NER results.

phenomenon further.

5.5 Related Work

[1] present a semi-supervised learning algorithm called alternating structure optimization (ASO). They find a low-dimensional projection of the input features that gives good linear classifiers over auxiliary tasks. These auxiliary tasks are sometimes specific to the supervised task, and sometimes general language modeling tasks like "predict the missing word". [135] present a semi-supervised extension of CRFs. (In [136], they extend their semi-supervised approach to more general conditional models.) One of the advantages of the semi-supervised learning approach that we use is that it is simpler and more general than that of

[1] and [135]. Their methods dictate a particular choice of model and training regime and could not, for instance, be used with an NLP system based upon an SVM classifier.

[81] present a K-means-like non-hierarchical clustering algorithm for phrases, which uses MapReduce. Since they can scale to millions of phrases, and they train over 800B unlabeled words, they achieve state-of-the-art accuracy on NER using their phrase clusters. This suggests that extending word representations to phrase representations is worth further investigation.

5.6 Conclusions

Word features can be learned in advance in an unsupervised, task-inspecific, and model-agnostic manner. These word features, once learned, are easily disseminated with other researchers, and easily integrated into existing supervised NLP systems. The disadvantage, however, is that accuracy might not be as high as a semi-supervised method that includes task-specific information and that jointly learns the supervised and unsupervised tasks [1; 135; 136].

Unsupervised word representations have been used in previous NLP work, and have demonstrated improvements in generalization accuracy on a variety of tasks. Ours is the first work to systematically compare different word representations in a controlled way. We found that Brown clusters and word embeddings both can improve the accuracy of a near-state-of-the-art supervised NLP system. We also found that combining different word representations can improve accuracy further. Error analysis indicates that Brown clustering induces better representations for rare words than C&W embeddings that have not received many training updates.

Another contribution of our work is a default method for setting the scaling parameter for word embeddings. With this contribution, word embeddings can now be used off-the-shelf as word features, with no tuning.

Future work should explore methods for inducing phrase representations, as well as techniques for increasing in accuracy by using word representations in *compound* features.

Replicating our experiments

You can visit `http://metaoptimize.com/projects/wordreprs/` to find: The word representations we induced, which you can download and use in your experiments; The code for inducing the word representations, which you can use to induce word representations on your own data; The NER and chunking system, with code for replicating our experiments.

Chapter 6

Disambiguation to Wikipedia

Disambiguating concepts and entities in a context sensitive way is a fundamental problem in natural language processing. The comprehensiveness of Wikipedia has made the online encyclopedia an increasingly popular target for disambiguation. Recently, disambiguation to Wikipedia (D2W) has been shown to form a valuable component for numerous natural language processing tasks including text classification [51; 21], measuring semantic similarity between texts [50], cross-document co-reference resolution [44; 87], and other tasks [74].

D2W is similar to a traditional Word Sense Disambiguation task, but distinct in that the Wikipedia link structure provides additional information about which disambiguations are compatible. In this section we analyze approaches that utilize this information to arrive at coherent sets of disambiguations for a given document (which we call "global" approaches), and compare them to more traditional (local) approaches. *Local* D2W approaches disambiguate each mention in a document separately, utilizing clues such as the textual similarity between the document and each candidate disambiguation's Wikipedia page. Recent work on D2W has tended to focus on more sophisticated *global* approaches to the problem, in which all mentions in a document are disambiguated simultaneously to arrive at a *coherent* set of disambiguations [29; 93; 162]. For example, if a mention of "Michael Jordan" refers to the computer scientist rather than the basketball player, then we would expect a mention of "Monte Carlo" in the same document to refer to the statistical technique rather than the location. Global approaches utilize the Wikipedia link graph to estimate coherence.

6.1 Contributions

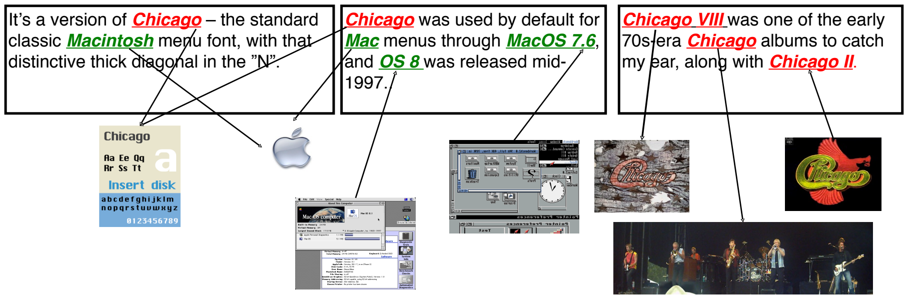
We show that previous approaches for global disambiguation can be improved, but even then the local disambiguation provides a baseline which is very hard to beat. Our contributions are as follows:

- We present a formulation of the D2W task as an optimization problem with local and global variants, and identify the strengths and the weaknesses of each.

- Using this formulation, we present a new global D2W system, called GLOW. In experiments on existing and novel D2W data sets, GLOW is shown to outperform the previous state-of-the-art system of [93]

- We present an error analysis and identify the key remaining challenge: determining when mentions refer to concepts *not* captured in Wikipedia.

6.2 Problem Definition and Approach



It's a version of ***Chicago*** – the standard classic ***Macintosh*** menu font, with that distinctive thick diagonal in the "N".

Chicago was used by default for ***Mac*** menus through ***MacOS 7.6***, and ***OS 8*** was released mid-1997.

Chicago VIII was one of the early 70s-era ***Chicago*** albums to catch my ear, along with ***Chicago II***.

Figure 6.1: An example of a Wikification system output.

We formalize our *Disambiguation to Wikipedia* (D2W) task as follows. We are given a document d with a set of mentions $M = \{m_1, \ldots, m_N\}$, and our goal is to produce a mapping from the set of mentions to the set of Wikipedia titles $W = \{t_1, \ldots, t_{|W|}\}$. Often, mentions correspond to a concept *without* a Wikipedia page; we treat this case by adding a special *null* title to the set W. Figure 6.1 shows sample output of a Wikification system on three input documents.

The D2W task can be visualized as finding a many-to-one matching on a bipartite graph, with mentions forming one partition and Wikipedia titles the other (see Figure 6.2). We denote the output matching as an N-tuple $\Gamma = (t_1, \ldots, t_N)$ where t_i is the output disambiguation for mention m_i.

A *local* D2W approach disambiguates each mention m_i separately. Specifically, let $\phi(m_i, t_j)$ be a score function reflecting the likelihood that the candidate title $t_j \in W$ is the correct disambiguation for $m_i \in M$. A local approach solves the following optimization problem:

$$\Gamma^*_{\text{local}} = \arg\max_{\Gamma} \sum_{i=1}^{N} \phi(m_i, t_i) \tag{6.1}$$

Local D2W approaches, exemplified by [15] and [90], utilize ϕ functions that assign higher scores to

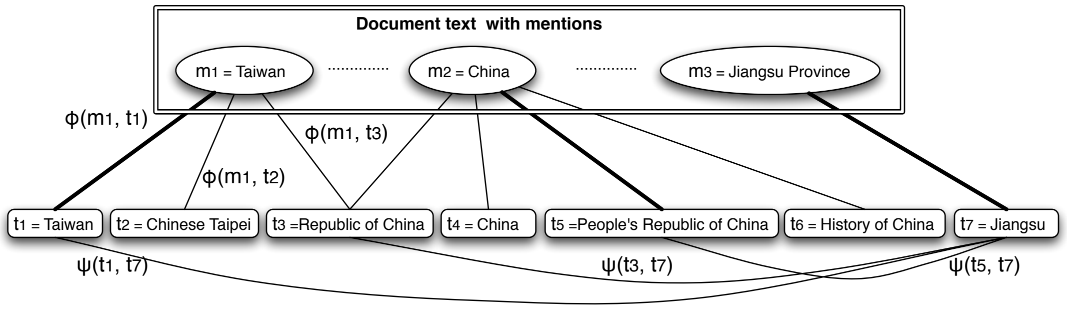


Figure 6.2: Sample Disambiguation to Wikipedia problem with three mentions. The mention "Jiangsu" is unambiguous. The correct mapping from mentions to titles is marked by heavy edges

titles with content similar to that of the input document.

We expect, all else being equal, that the correct disambiguations will form a "coherent" set of related concepts. Global approaches define a coherence function ψ, and attempt to solve the following disambiguation problem:

$$\Gamma^* = \arg\max_{\Gamma}[\sum_{i=1}^{N} \phi(m_i, t_i) + \psi(\Gamma)] \tag{6.2}$$

The global optimization problem in Equation 6.2 is NP-hard, and approximations are required [29]. The common approach is to utilize the Wikipedia link graph to obtain an estimate pairwise relatedness between titles $\psi(t_i, t_j)$ and to efficiently generate a *disambiguation context* Γ', a rough approximation to the optimal Γ^*. We then solve the easier problem:

$$\Gamma^* \approx \arg\max_{\Gamma} \sum_{i=1}^{N}[\phi(m_i, t_i) + \sum_{t_j \in \Gamma'} \psi(t_i, t_j)] \tag{6.3}$$

Eq. 6.3 can be solved by finding each t_i and then mapping m_i independently as in a local approach, but still enforces some degree of coherence among the disambiguations.

6.3 System Architecture

In this section, we present our global D2W system, which solves the optimization problem in Equation 6.3. We refer to the system as GLOW, for Global Wikification. We use GLOW as a test bed for evaluating local and global approaches for D2W. GLOW combines a powerful local model ϕ with an novel method for choosing an accurate disambiguation context Γ', which as we show in our experiments allows it to outperform the previous state of the art.

```
Algorithm: Disambiguate to Wikipedia
Input: document $d$, Mentions $M = \{m_1, \ldots, m_N\}$
Output: a disambiguation $\Gamma = (t_1, \ldots, t_N)$.
1) Let $M'$ = $M \cup$ { Other potential mentions in $d$}
2) For each mention $m'_i \in M'$, construct a set of disambiguation candidates $T_i = \{t^i_1, \ldots, t^i_{k_i}\}, t^i_j \neq$ null
3) Ranker: Find a solution $\Gamma = (t'_1, \ldots, t'_{|M'|})$, where $t'_i \in T_i$ is the best non-null disambiguation of $m'_i$.
4) Linker: For each $m'_i$, map $t'_i$ to null in $\Gamma$ iff doing so improves the objective function 6.3
5) Return $\Gamma$ entries for the original mentions $M$.
```

Figure 6.3: High-level pseudocode for GLOW.

We represent the functions ϕ and ψ as weighted sums of features. Specifically, we set:

$$\phi(m,t) = \sum_i w_i \phi_i(m,t) \tag{6.4}$$

where each feature $\phi_i(m,t)$ captures some aspect of the relatedness between the mention m and the Wikipedia title t. Feature functions $\psi_i(t,t')$ are defined analogously. We detail the specific feature functions utilized in GLOW in following sections. The coefficients w_i are learned using a Support Vector Machine over bootstrapped training data from Wikipedia, as described in Section 6.7.

At a high level, the GLOW system optimizes the objective function in Eq. 6.3 in a two-stage process. We first execute a *ranker* to obtain the best non-null disambiguation for each mention in the document, and then execute a *linker* that decides whether the mention should be linked to Wikipedia, or whether instead switching the top-ranked disambiguation to *null* improves the objective function. As our experiments illustrate, the linking task is the more challenging of the two by a significant margin.

Figure 6.3 provides detailed pseudocode for GLOW. Given a document d and a set of mentions M, we start by augmenting the set of mentions with all phrases in the document that *could* be linked to Wikipedia, but were not included in M. Introducing these additional mentions provides context that may be informative for the global coherence computation (it has no effect on local approaches). In the second step, we construct for each mention m_i a limited set of candidate Wikipedia titles T_i that m_i may refer to. Considering only a small subset of Wikipedia titles as potential disambiguations is crucial for tractability (we refer the reader for the details to [111]). In the third step, the ranker outputs the most appropriate non-null disambiguation t_i for each mention m_i.

In the final step, the linker decides whether the top-ranked disambiguation is correct. The disambiguation (m_i, t_i) may be incorrect for several reasons: (1) mention m_i does not have a corresponding Wikipedia page, (2) m_i does have a corresponding Wikipedia page, but it was not included in T_i, or (3) the ranker

Baseline Feature: $P(t\|m)$, $P(t)$
Local Features: $\phi_i(t,m)$
cosine-sim(Text(t),Text(m)) : Naive/Reweighted
cosine-sim(Text(t),Context(m)): Naive/Reweighted
cosine-sim(Context(t),Text(m)): Naive/Reweighted
cosine-sim(Context(t),Context(m)): Naive/Reweighted
Global Features: $\psi_i(t_i,t_j)$
$I_{[t_i-t_j]}$*PMI(InLinks(t_i),InLinks(t_j)) : avg/max*
$I_{[t_i-t_j]}$*NGD(InLinks(t_i),InLinks(t_j)) : avg/max*
$I_{[t_i-t_j]}$*PMI(OutLinks(t_i),OutLinks(t_j)) : avg/max*
$I_{[t_i-t_j]}$*NGD(OutLinks(t_i),OutLinks(t_j)) : avg/max*
$I_{[t_i\leftrightarrow t_j]}$ *: avg/max*
$I_{[t_i\leftrightarrow t_j]}$*PMI(InLinks($t_i$),InLinks($t_j$)) : avg/max*
$I_{[t_i\leftrightarrow t_j]}$*NGD(InLinks($t_i$),InLinks($t_j$)) : avg/max*
$I_{[t_i\leftrightarrow t_j]}$*PMI(OutLinks($t_i$),OutLinks($t_j$)) : avg/max*
$I_{[t_i\leftrightarrow t_j]}$*NGD(OutLinks($t_i$),OutLinks($t_j$)) : avg/max*

Table 6.1: Ranker features. $I_{[t_i-t_j]}$ is an indicator variable which is 1 iff t_i links to t_j or vise-versa. $I_{[t_i\leftrightarrow t_j]}$ is 1 iff the titles point to each other.

erroneously chose an incorrect disambiguation over the correct one.

In the below sections, we describe each step of the GLOW algorithm, and the local and global features utilized, in detail. Because we desire a system that can process documents at scale, each step requires trade-offs between accuracy and efficiency.

6.4 Local Features

Following previous work, we use Wikipedia hyperlinks to perform these steps. GLOW utilizes an *anchor-title* index, computed by crawling Wikipedia, that maps each distinct hyperlink anchor text to its target Wikipedia titles. For example, the anchor text "Chicago" is used in Wikipedia to refer both to the city in Illinois and to the movie. From the anchor-title index, we compute two local features $\phi_i(m,t)$. The first, $P(t|m)$, is the fraction of times the title t is the target page for an anchor text m. This single feature is a very reliable indicator of the correct disambiguation [42], and we use it as a baseline in our experiments. The second, $P(t)$, gives the fraction of all Wikipedia articles that link to t.

In addition to the two above baseline features, we compute a set of text-based local features $\phi(t,m)$. These features capture the intuition that a given Wikipedia title t is more likely to be referred to by mention m appearing in document d if the Wikipedia page for t has high textual similarity to d, or if the context surrounding hyperlinks to t are similar to m's context in d.

For each Wikipedia title t, we construct a top-200 token TF-IDF summary of the Wikipedia page t, which we denote as $Text(t)$ and a top-200 token TF-IDF summary of the context within which t was hyperlinked to in Wikipedia, which we denote as $Context(t)$. We keep the IDF vector for all tokens in Wikipedia, and given an input mention m in a document d, we extract the TF-IDF representation of d, which we denote $Text(d)$, and a TF-IDF representation of a 100-token window around m, which we denote $Context(m)$. This allows us to define four local features described in Table 6.1.

We additionally compute *weighted* versions of the features described above. Error analysis has shown that in many cases the summaries of the different disambiguation candidates for the same surface form s were very similar. For example, consider the disambiguation candidates of "China" and their TF-IDF summaries in Figure 6.2. The majority of the terms selected in *all* summaries refer to the general issues related to China, such as *"legalism, reform, military, control, etc."*, while a minority of the terms actually allow disambiguation between the candidates. The problem stems from the fact that the TF-IDF summaries are constructed against the entire Wikipedia, and not against the confusion set of disambiguation candidates of m. Therefore, we *re-weigh* the TF-IDF vectors using the TF-IDF scheme on the disambiguation candidates as a ad-hoc document collection, similarly to an approach in [139] for classifying documents. In our scenario, the TF of the a token is the original TF-IDF summary score (a real number), and the IDF term is the sum of all the TF-IDF scores for the token within the set of disambiguation candidates for m. This adds 4 more "re-weighted local" features in Table 6.1.

6.5 Global Features

Global approaches require a disambiguation context Γ' and a relatedness measure ψ in Equation 6.3. In this section, we describe our method for generating a disambiguation context, and the set of global features $\psi_i(t, t')$ forming our relatedness measure.

In previous work, Cucerzan defined the disambiguation context as the union of disambiguation candidates for all the named entity mentions in the input document [29]. The disadvantage of this approach is that irrelevant titles are inevitably added to the disambiguation context, creating noise. Milne and Witten, on the other hand, use a set of unambiguous mentions [93]. This approach utilizes only a fraction of the available mentions for context, and relies on the presence of unambiguous mentions with high disambiguation utility. In GLOW, we utilize a simple and efficient alternative approach: we first train a local disambiguation system, and then use the predictions of that system as the disambiguation context. The advantage of this approach is that unlike [93] we use all the available mentions in the document, and unlike [29] we reduce the amount of irrelevant titles in the disambiguation context by taking only the top-ranked disambiguation per mention.

Our global features are refinements of previously proposed semantic relatedness measures between Wikipedia titles. We are aware of two previous methods for estimating the relatedness between two Wikipedia concepts: [133], which uses category overlap, and [92], which uses the incoming link structure. Previous work experimented with two relatedness measures: NGD, and Specificity-weighted Cosine Similarity. Consistent with previous work, we found NGD to be the better-performing of the two. Thus we use only NGD along with a well-known Pointwise Mutual Information (PMI) relatedness measure. Given a Wikipedia title collection W, titles t_1 and t_2 with a set of incoming links L_1, and L_2 respectively, PMI and NGD are defined as follows:

$$NGD(L_1, L_2) = \frac{Log(Max(|L_1|, |L_2|)) - Log(|L_1 \cap L_2|)}{Log(|W|) - Log(Min(|L_1|, |L_2|))}$$

$$PMI(L_1, L_2) = \frac{|L_1 \cap L_2|/|W|}{|L_1|/|W||L_2|/|W|}$$

The NGD and the PMI measures can also be computed over the set of *outgoing* links, and we include these as features as well. We also included a feature indicating whether the articles each link to one another. Lastly, rather than taking the sum of the relatedness scores as suggested by Equation 6.3, we use two features: the average and the maximum relatedness to Γ'. We expect the average to be informative for many documents. The intuition for also including the maximum relatedness is that for longer documents that may cover many different subtopics, the maximum may be more informative than the average.

We have experimented with other semantic features, such as category overlap or cosine similarity between the TF-IDF summaries of the titles, but these did not improve performance in our experiments. The complete set of global features used in GLOW is given in Table 6.1.

6.6 Linker Features

Given the mention m and the top-ranked disambiguation t, the linker attempts to decide whether t is indeed the correct disambiguation of m. The linker includes the same features as the ranker, plus additional features we expect to be particularly relevant to the task. We include the confidence of the ranker in t with respect to second-best disambiguation t', intended to estimate whether the ranker may have made a mistake. We also include several properties of the mention m: the entropy of the distribution $P(t|m)$, the percent of Wikipedia titles in which m appears hyperlinked versus the percent of times m appears as plain text, whether m was detected by NER as a named entity, and a Good-Turing estimate of how likely m is to be out-of-Wikipedia concept based on the counts in $P(t|m)$.

Mentions/Distinct titles			
data set	Gold	Identified	Solvable
ACE	257/255	213/212	185/184
MSNBC	747/372	530/287	470/273
AQUAINT	727/727	601/601	588/588
Wikipedia	928/813	855/751	843/742

Table 6.2: Number of mentions and corresponding distinct titles by data set. Listed are (number of mentions)/(number of distinct titles) for each data set, for each of three mention types. *Gold* mentions include all disambiguated mentions in the data set. *Identified* mentions are gold mentions whose correct disambiguations exist in GLOW's author-title index. *Solvable* mentions are identified mentions whose correct disambiguations are among the candidates selected by GLOW (see Table 3).

6.7 Linker and Ranker Training

We train the coefficients for the ranker features using a linear Ranking Support Vector Machine, using training data gathered from Wikipedia. Wikipedia links are considered gold-standard links for the training process. The methods for compiling the Wikipedia training corpus are given in Section 6.8.

We train the linker as a separate linear Support Vector Machine. Training data for the linker is obtained by applying the ranker on the training set. The mentions for which the top-ranked disambiguation did not match the gold disambiguation are treated as negative examples, while the mentions the ranker got correct serve as positive examples.

6.8 Datasets

We evaluate GLOW on four data sets, of which two are from previous work. The first data set, from [93], is a subset of the *AQUAINT* corpus of newswire text that is annotated to mimic the hyperlink structure in Wikipedia. That is, only the first mentions of "important" titles were hyperlinked. Titles deemed uninteresting and redundant mentions of the same title are not linked. The second data set, from [29], is taken from *MSNBC* news and focuses on disambiguating named entities after running NER and co-reference resolution systems on newsire text. In this case, *all* mentions of all the detected named entities are linked.

We also constructed two additional data sets. The first is a subset of the *ACE* co-reference data set, which has the advantage that mentions and their types are given, and the co-reference is resolved. We asked annotators on Amazon's Mechanical Turk to link the first nominal mention of each co-reference chain to Wikipedia, if possible. Finding the accuracy of a majority vote of these annotations to be approximately 85%, we manually corrected the annotations to obtain ground truth for our experiments. The second data set we constructed, *Wiki*, is a sample of paragraphs from Wikipedia pages. Mentions in this data set correspond to existing hyperlinks in the Wikipedia text. Because Wikipedia editors explicitly link mentions to Wikipedia pages, their anchor text tends to match the title of the linked-to-page—as a result, in the overwhelming majority of cases, the disambiguation decision is as trivial as string matching. In an attempt to generate more challenging data, we extracted 10,000 random paragraphs for which choosing the top disambiguation according to $P(t|m)$ results in at least a 10% ranker error rate. 40 paragraphs of this data was utilized for testing, while the remainder was used for training.

The data sets are summarized in Table 6.2. The table shows the number of annotated mentions which were hyperlinked to *non-null* Wikipedia pages, and the number of titles in the documents (without counting repetitions). For example, the AQUAINT data set contains 727 mentions,[1] all of which refer to distinct titles. The MSNBC data set contains 747 mentions mapped to non-null Wikipedia pages, but some mentions within the same document refer to the same titles. There are 372 titles in the data set, when multiple instances of the same title within one document are not counted.

6.9 Evaluation Methodology

To isolate the performance of the individual components of GLOW, we use multiple distinct metrics for evaluation. *Ranker accuracy*, which measures the performance of the ranker alone, is computed only over those mentions with a non-null gold disambiguation that appears in the candidate set. It is equal to the fraction of these mentions for which the ranker returns the correct disambiguation. Thus, a perfect ranker should achieve a ranker accuracy of 1.0, irrespective of limitations of the candidate generator. *Linker accuracy* is defined as the fraction of *all* mentions for which the linker outputs the correct disambiguation (note that, when the title produced by the ranker is incorrect, this penalizes linker accuracy). Lastly, we evaluate our whole system against other baselines using a previously-employed *"bag of titles" (BOT)* evaluation [93], [111]. In *BOT*, we compare the set of titles output for a document with the gold set of titles for that document (ignoring duplicates), and utilize standard precision, recall, and F1 measures.

6.10 Experiments and Results

In this section, we evaluate and analyze GLOW's performance on the D2W task. We begin by evaluating the mention detection component (Step 1 of the algorithm). The second column of Table 6.2 shows how many of the "non-null" mentions and corresponding titles we could successfully identify (e.g. out of 747 mentions

[1]The data set contains votes on how important the mentions are. We believe that the results in [93] were reported on mentions which the majority of annotators considered important. In contrast, we used all the mentions.

	Data sets			
Features	ACE	MSNBC	AQUAINT	Wiki
$P(t\|m)$	94.05	81.91	93.19	85.88
$P(t\|m)$+Local				
Naive	95.67	84.04	94.38	92.76
Reweighted	96.21	85.10	**95.57**	**93.59**
All above	95.67	84.68	95.40	93.59
$P(t\|m)$+Global				
NER	96.21	84.04	94.04	89.56
Unambiguous	94.59	84.46	95.40	89.67
Predictions	**96.75**	**88.51**	**95.91**	89.79
$P(t\|m)$+Local+Global				
All features	**97.83**	**87.02**	94.38	**94.18**

Table 6.3: Ranker Accuracy. Bold values indicate the best performance in each feature group. The global approaches marginally outperform the local approaches on *ranker accuracy* , while combing the approaches leads to further marginal performance improvement.

Data set	Local	Global	Local+Global
ACE	$80.1 \rightarrow 82.8$	$80.6 \rightarrow 80.6$	$81.5 \rightarrow 85.1$
MSNBC	$74.9 \rightarrow 76.0$	$77.9 \rightarrow 77.9$	$76.5 \rightarrow 76.9$
AQUAINT	$93.5 \rightarrow 91.5$	$93.8 \rightarrow 92.1$	$92.3 \rightarrow 91.3$
Wiki	$92.2 \rightarrow 92.0$	$88.5 \rightarrow 87.2$	$92.8 \rightarrow 92.6$

Table 6.4: Linker performance. The notation $X \rightarrow Y$ means that when linking all mentions, the linking accuracy is X, while when applying the trained linker, the performance is Y. The local approaches are better suited for linking than the global approaches. The linking accuracy is very sensitive to domain changes.

in the MSNBC data set, only 530 appeared in our anchor-title index). Missing entities were primarily due to especially rare surface forms, or sometimes due to idiosyncratic capitalization in the corpus. Improving the number of identified mentions substantially is non-trivial; [165] managed to successfully identify only 59 more entities than we do in the MSNBC data set, using a much more powerful detection method based on search engine query logs.

Next, we evaluate the accuracy of the ranker. Table 6.3 compares the ranker performance with baseline, local and global features. The reweighted local features outperform the unweighted ("Naive") version, and the global approach outperforms the local approach on all data sets except Wikipedia. As the table shows, our approach of defining the disambiguation context to be the predicted disambiguations of a simpler local model ("Predictions") performs better than using NER entities as in [29], or only the unambiguous entities as in [93].[2] Combining the local and the global approaches typically results in minor improvements.

While the global approaches are most effective for ranking, the linking problem has different characteristics as shown in Table 6.4. We can see that the global features are not helpful in general for predicting

[2]In NER we used only the top prediction, because using all candidates as in [29] proved prohibitively inefficient.

System	ACE	MSNBC	AQUAINT	Wiki
Baseline: $P(t\|m)$	69.52	72.83	82.67	81.77
GLOW Local	75.60	74.39	84.52	90.20
GLOW Global	74.73	74.58	84.37	86.62
GLOW	77.25	74.88	83.94	90.54
M&W	72.76	68.49	83.61	80.32

Table 6.5: End systems performance - BOT F1. The performance of the full system (GLOW) is similar to that of the local version. GLOW outperforms [93] on all data sets.

whether the top-ranked disambiguation is indeed the correct one.

Further, although the trained linker improves accuracy in some cases, the gains are marginal—and the linker decreases performance on some data sets. One explanation for the decrease is that the linker is trained on Wikipedia, but is being tested on non-Wikipedia text which has different characteristics. However, in separate experiments we found that training a linker on out-of-Wikipedia text only increased test set performance by approximately 3 percentage points. Clearly, while ranking accuracy is high overall, different strategies are needed to achieve consistently high linking performance.

A few examples from the ACE data set help illustrate the tradeoffs between local and global features in GLOW. The global system mistakenly links *"<Dorothy Byrne>, a state coordinator for the Florida Green Party, said . . . "* to the British journalist, because the journalist sense has high coherence with other mentions in the newswire text. However, the local approach correctly maps the mention to $null$ because of a lack of local contextual clues. On the other hand, in the sentence *"Instead of Los Angeles International, for example, consider flying into <Burbank> or John Wayne Airport in Orange County, Calif."*, the local ranker links the mention *Burbank* to *Burbank,_California*, while the global system correctly maps the entity to *Bob_Hope_Airport*, because the three airports mentioned in the sentence are highly related to one another.

Lastly, in Table 6.5 we compare the end system BOT F1 performance. The local approach proves a very competitive baseline which is hard to beat. Combining the global and the local approach leads to marginal improvements. The full GLOW system outperforms the existing state-of-the-art system from [93], denoted as M&W, on all data sets. An important question is why M&W underperforms the baseline on the MSNBC and Wikipedia data sets. In an error analysis, M&W performed poorly on the MSNBC data not due to poor disambiguations, but instead because the data set contains only named entities, which were often delimited incorrectly by M&W. Wikipedia was challenging for a different reason: M&W performs less well on the short (one paragraph) texts in that set, because they contain relatively few of the unambiguous entities the system relies on for disambiguation.

6.11 TAC Entity Linking

Traditional information extraction evaluations, such as the Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE), assess the ability to extract information from individual documents in isolation. In practice, however, we may need to gather information about a person or organization that is scattered among the documents of a large collection. This requires the ability to identify the relevant documents and to integrate facts, possibly redundant, possibly complementary, possibly in conflict, coming from these documents. Furthermore, we may want to use the extracted information to augment an existing data base. This requires the ability to link individuals mentioned in a document, and information about these individuals, to entries in the data base.

The Knowledge Base Population (KBP) shared task was established to address and evaluate these capabilities. This is done through two separate subtasks, Entity Linking and Slot Filling. For both tasks, the system is given a name and a document in which this name appears. For Entity Linking, it must decide whether this name corresponds to an entry in a data base and, if so, which one. For Slot Filling, the system must determine from a large corpus the values of specified attributes of the entity, such as the age and birthplace of a person or the top employees of a corporation. We wanted to evaluate the performance of our Wikification system [111] by competing in the 2011 KBP Entity Linking task. While the task is somewhat different from Wikification, it is similar enough to shed light on the strengths and the weaknesses of our Wikifier.

The Entity Linking task in KBP is formalized as follows. The organizers provide a set $KB = \{E_1, E_2, \ldots, E_{|KB|}\}$ of reference entities E_i, which is a subset of Wikipedia[3]. The organizers also provide a list of queries, which are tripples of the form $(Q_{id}, Q_{form}, Q_{text})$, where Q_{id} is a reference number for the query, Q_{form} is the surface form of the query, and Q_{text} is the context within which the surface form appears. Not each query can be mapped to KB or to Wikipedia. For those queries which can be mapped, the goal of the entity-linking task is to provide the mapping to KB. For those queries which cannot be mapped to KB, as of 2011, the goal is to cluster together the queries which refer to the same entity. Therefore, the TAC Entity Linking task is a combination of a Wikification-like task and cross-document co-reference resolution-like task.

We note that the structure of the queries makes the task somewhat different than traditional Wikification or cross-document co-reference resolution. The subtlety is that the surface form of the query is not always the "canonical" representation of the queried entity in the text. We clarify this point in Table 6.6 and show sample queries from the 2011 evaluation dataset. We note that for the query *EL_02067*, if we could resolve

[3]The TAC knowledge base contains 818,741 reference entities, which is about a third of 2009 Wikipedia pages.

Query ID	*EL_02067*	*EL_02243*
Form	*Titans*	*Zanesville*
Text	*"Tenn. - Jeff Fisher issued his most positive report yesterday on Vince Young's status, but the <u>Tennessee Titans</u>$[m_1]$ coach still is not prepared to declare that the quarterback is to start tomorrow's AFC wild-card playoff game against the Chargers. Young is dealing with a right quadriceps injury. "He's better, like I anticipated", Fisher said after the <u>Titans</u>$[m_2]$ had worked at their indoor facility... The Tennessean, a Nashville newspaper, is polling those who tap in concerning who should start at quarterback for the <u>Titans</u>$[m_3]$, Young or Collins. At midday yesterday, Young led 56 percent to 44 percent. For a time, Collins had led."*	*OBAMA SAYS HE WOULD ENHANCE SOCIAL SERVICES VIA RELIGIOUS GROUPS Jeff Zeleny reported from <u>Zanesville</u>$[m_1]$, and Michael Luo from New York. As he announced his proposal on Tuesday, standing outside the Eastside Community Ministry in <u>Zanesville</u>$[m_2]$, Obama harked back to his early days as a community organizer in Chicago, where Catholic charities financed his programs... <u>ZANESVILLE</u>$[m_3]$*, Ohio.

Table 6.6: Sample TAC Entity Linking queries.

```
Given a knowledge base $KB$ and a query $(Q_{id}, Q_{form}, Q_{text})$
  1. Select a set of reference mentions $R = \{m_1, \ldots, m_N\}$ in $Q_{text}$. The set of reference mentions
     are basically anchors in the text which are likely to be co-referent with the $Q_{form}$ in the
     context of the text. For example, in Table 6.6, { Tennessee Titans$[m_1]$, Titans$[m_2]$, Titans$[m_3]$ }
     are our sample reference mentions for the query EL_02067.
  2. Assign to the set of reference mentions $R$ a set of disambiguations $D = \{(m_i, t_i, c_i)\}_{i=1}^N$. For
     each reference mention $m_i$, $t_i$ is the Wikipedia title assigned to $m_i$, and $c_i$ is the confidence
     of the disambiguation.
  3. Given a set $D$ of possibly inconsistent disambiguations, decide on a single disambiguation
     for $Q_{id}$ to $KB$, or map to $NULL$ if no mapping can be found.
```

Figure 6.4: Pseudocode for the Illinois entity linking system in TAC 2011.

the within-document co-references *"Titans"-"<u>Tennessee Titans</u>$[m_1]$"*, the entity linking task would become trivial. The same applies for the query *EL_02243* with the co-reference *"Zanesville"- "<u>ZANESVILLE</u>$[m_3]$, Ohio."*.

The KBP Entity Linking dataset was generated to respect one sense per document assumption, therefore, whenever the query surface form (e.g. *Titans*) apepars in the document, we can assume it refers to the same entity. This property of the entity linking task creates an opportunity for *query expansion*, which make the task somewhat different from the traditional Wikification or cross-document co-reference resolution.

6.11.1 Entity Linking Approach

We focus only on the "Wikification" part of the task. For the queries which cannot be linked to the knowledge base, we provide a trivial solution for cross-document co-reference resolution by clustering all the queries with identical surface forms together. In Figure 6.4 we outline the pseudocode for the Illinois entity linking system in TAC 2011. The system has three main components: reference mention generation which we describe in detail in Section 6.11.2, reference mention disambiguation which we describe in detail in Section 6.11.3, and assembling the resulting disambiguations to a single solution which we describe in Section 6.11.4. In Section 6.11.5 we report the experimental results and conclude.

6.11.2 Reference Mentions Selection

This step is very similar to query expansion as described, for example in [22]. The difference is that unlike the traditional query expansion, the reference mentions will be bound to specific locations in the text. For example, we mark the reference mentions set { *Tennessee Titans*$[m_1]$, *Titans*$[m_2]$, *Titans*$[m_3]$ } in Table 6.6. We note that we will disambiguate the mentions jointly, however we will ultimately allow each reference mention have a different disambiguation. Therefore, theoretically, in our system *Titans*$[m_2]$ and *Titans*$[m_3]$ could have different disambiguations. We believe that this architecture is more robust since it allows us to be more flexible in suggesting reference mentions for the query form, and to recover from potentially erroneous reference mentions. This architecture also allows us to be robust to documents which do not have a "one sense per document" property.

Below we describe our heuristic for reference mention recommendation.

1. Annotate Q_{text} with Illinois NER tagger [112][4]. Let $NER_{Q_{form}}(Q_{text})$ be the set of all named entities in Q_{text} which could be matched through approximate string matching to Q_{form}. Approximate string matching we applied was acronym matching (for example, *AI* would be matched with *Artificial Intelligence*) and simple rules for matching named entities, which allowed matching *Mr. Bush* to *GEORGE W. BUSH*. That is, we have simple rules for discarding professional titles, honorifics, case-insensitive matching and punctuation-insensitive matching.

2. If $NER_{Q_{form}}(Q_{text}) \neq \emptyset$, let CF be the longest string in $NER_{Q_{form}}(Q_{text})$, let $CF = Q_{form}$ otherwise. The purpose of this step is to identify the *"canionical form"* (CF) for the query form in the text. For example, in Table 6.6, { *Tennessee Titans*$[m_1]$ is the canonical form for the query *EL_02067*, and { *Zanesville*$[m_1]$ is the canonical form for the query *EL_02243*.

[4]Available at `http://cogcomp.cs.illinois.edu/page/software`

3. The Wikification system of [111] does not perform an approximate string matching, it cross-links only the expressions which appeared as hyperlinks in Wikipedia. Therefore, the system of [111] would not be able to cross-link the string ZANESVILLE to Wikipedia. Intuitively, we want to normalize *ZANESVILLE* to *Zanesville* and *Sharm Elsheikh* to *Sharm el-Sheikh*. This step is particularly useful for named entities with non-English origin. For example it Sharm el-Sheikh has multiple transliterations, including "Sharm El Sheikh", "Sharm Al Sheikh", 'Sharm AlSheikh", 'Sharm al-Sheikh" etc. It may be useful to pre-process the query text to make it easier for the Wikification system to disambiguate the reference mentions to Wikipedia. In this step, we normalize the canonical form CN to the normalized canonical form NCF. Following [90], we define *linkability* of an expression s as the ratio of Wikipedia pages which contain s as a hyperlink anchor to the number of Wikipedia pages which contain s in any form. For example, 1154 Wikipedia pages contain the expression *"Michael Jordan"* and out of them 959 (83%) also contain a link anchored as *"Michael Jordan"* to the Wikipedia page corresponding to the correct meaning. In constrast the expression "boarding school" appeared in 5038 Wikipedia pages, and only 1421 (28%) of them, had a hyperlink anchored with the surface from. To obtain the canonical normalized surface form for CF we compare CF against the list of all titles, redirects and hyperlink anchors in Wikipedia. We keep only those, which appeared in at least 10 Wikipedia pages and can be matched using a case-insensitive, punctuation-insensitive approximate string matching heuristis. Among the set of matched expressions, we choose the most *linkable* one.

4. If $CF \neq Q_{form}$ and CF cannot be matched to neither Wikipedia anchors nor titles nor redirects, we are assuming that an NER error has occurred and we revert to the normalization step with $CF = Q_{form}$.

5. We replace all the instances of CF in Q_{text} by NCF, and mark these instances as out final set of reference mentions. The Wikifier of [111] will be applied to this modified text.

6.11.3 Disambiguation

The disambiguation component of our entity linking system is performed through a straighforward application of the GLOW Wikification system of [111] described in Chapter 6. In this section we provide a short summary of GLOW .

We formalize the task as finding a many-to-one matching on a bipartite graph, with mentions forming one partition and Wikipedia titles the other (see Figure 6.5). We denote the output matching as an N-tuple $\Gamma = (t_1, \ldots, t_N)$ where t_i is the output disambiguation for mention m_i. With this formulation in mind,

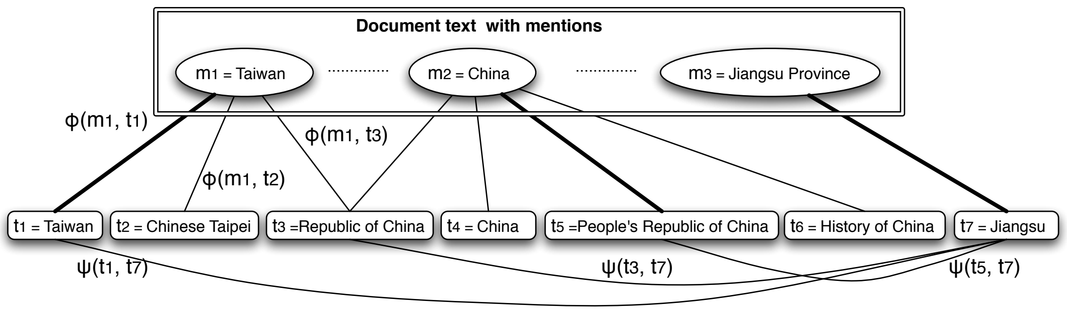


Figure 6.5: Sample disambiguation to Wikipedia with three mentions formalized as bipartite matching problem. The correct mapping from mentions to titles is marked by heavy edges

we can write down an objective function: The common approach is to utilize the Wikipedia link graph to obtain an estimate pairwise relatedness between titles $\psi(t_i, t_j)$ and to efficiently generate a *disambiguation context* Γ', a rough approximation to the optimal Γ^*. We then solve the easier problem:

$$\Gamma^* \approx \arg\max_{\Gamma} \sum_{i=1}^{N} [\phi(m_i, t_i) + \sum_{t_j \in \Gamma'} \psi(t_i, t_j)] \tag{6.5}$$

Where $\{m_i\}_{i=1}^N$ is the set of mentions, $\{t_i\}_{i=1}^N$ is a set of associated Wikipedia pages, ϕ is a local scoring function which assigns higher scores to titles with content similar to that of the input document, ψ is a coherence function which assigns higher scores to related titles in Wikipedia and Γ', a rough approximation to the optimal solution. We can solve the equation 6.5 efficiently by finding each t_i and then mapping m_i independently as in a local approach, but still enforces some degree of coherence among the disambiguations using Γ' and ψ.

The pseudocode for the original GLOW system is given in Figure 6.6. We note that the input document d and the mention set $M = \{m_1, \ldots, m_N\}$ which GLOW expects as input are the normalized text of Q_{text} and the set of reference mentions provided by the algorithm described in Section 6.11.2. We note that while we mark a specific set of mentions M for GLOW to link to Wikipedia, GLOW will identify and disambiguate other expressions in the input text as well, and use them as disambiguation context for disambiguating M.For our entity linking system, we disable the linker component (step 4 of GLOW). The reason is that we will form the final solution in Section 6.6.

```
**Algorithm: Disambiguate to Wikipedia**
Input: document $d$, Mentions $M = \{m_1, \dots, m_N\}$
Output: a disambiguation $\Gamma = (t_1, \dots, t_N)$.
1) Let $M'$ = $M \cup$ { Other potential mentions in $d$}
2) For each mention $m'_i \in M'$, construct a set of disambiguation candidates $T_i = \{t^i_1, \dots, t^i_{k_i}\}, t^i_j \neq$ null
3) Ranker: Find a solution $\Gamma = (t'_1, \dots, t'_{|M'|})$, where $t'_i \in T_i$ is the best non-null disambiguation of $m'_i$.
4*) Linker: For each $m'_i$, map $t'_i$ to null in $\Gamma$ iff doing so improves the objective function 6.5
5) Return $\Gamma$ entries for the original mentions $M$.
```

Figure 6.6: High-level pseudocode for GLOW . Step (4) is disabled for the entity-linking task.

6.11.4 Generating a Single Solution

The given an input of a document d and a set of mentions $M = \{m_1, \dots, m_N\}$ the GLOW system assigns each mention m_i a Wikipedia title t_i, and two confidence scores (r_i, l_i), where r_i is the *ranking confidence* that t_i is the most appropriate disambiguation among the disambiguation candidates proposed for m_i. l_i is the *linker score* indicating whether the objective function 6.5 would improve if we map m_i to $NULL$ instead of t_i. A positive score indicates that t_i is preferred over $NULL$ and a negative score indicates otherwise.

Given a knowledge base $KB = \{E_1, E_2, \dots, E_{|KB|}\}$, a query $(Q_{id}, Q_{form}, Q_{text})$ and a set of tuples $\{(m_i, t_i, r_i, l_i)\}_{1 \le i \le N}$ returned by GLOW , our goal is to assign a KB entry E^* to the query or $NULL$ if no such entry can be matched. Our first step is selecting a single Wikipedia page t^* out of the set $\{(m_i, t_i, r_i, l_i)\}$ (possibly $NULL$). We have explored four strategies:

1. **MaxNoThres** : Let
$$i^* = \arg\max_{1 \le i \le N} \{r_i\} \tag{6.6}$$
Then $t* = t_{i^*}$. The idea is very simple: just select a title t_{i^*} which was assigned the maximum ranker confidence.

2. **MaxWithThres** : Let
$$i^* = \arg\max_{1 \le i \le N \wedge l_i \ge 0} \{r_i\} \tag{6.7}$$
Then $t^* = t_{i^*}$. This strategy is identical to *MaxNoThres*, but we consider only the titles which were "linked" by GLOW. If GLOW assigned a negative linker score l_i to the mention m_i, we discard t_i from the list of possible results.

3. **SumNoThres** : Let
$$i^* = \arg\max_{1 \le i \le N} \sum_{t_j = t_i} r_j \tag{6.8}$$

Then $t* = t_{i^*}$. This strategy is similar to *MaxNoThres*, except we summarize the ranker scores for all the mentions mapped to the same title.

4. **SumWithThres** : Let

$$i^* = \underset{1 \le i \le N \wedge l_i \ge 0}{\arg\max} \sum_{t_j = t_i \wedge l_j \ge 0} r_j \tag{6.9}$$

Then $t^* = t_{i^*}$. This strategy is identical to *SumNoThres*, but we consider only the titles which were "linked" by GLOW. We also summarize only over "linked" mentions.

Once we mapped the query to a Wikipedia title t^*, our next step is to map t^* to an entry E^* in the KBP TAC knowledge base. Since GLOW and KBP can use different versions of Wikipedia, we used a very recent February 2011 version of Wikipedia redirects. Therefore, a Wikipedia title t matches the KBP TAC entry E if both redirect to the same page in the February 2011 version of Wikipedia.

6.11.5 Experiments and Results

It is important to note that the ranking and the linking components of GLOW are SVM models which have to be trained. In the results reported in [111], we trained the GLOW system on Wikipedia articles themselves, training the system to mimic the Wikipedia annotation scheme. For the TAC 2011 entity linking task, we have also trained a GLOW model on the three publicly available newswire Wikification datasets described in [111] as well as a collection on 97 blogs which we Wikified using the Mechanical Turk, but have not published yet. The annotation style wad consistent to the Wikification works such as [29] and [93] , it was not a TAC entity linking style annotation. Overall, our ″newswire″ training dataset contained 200 documents. Since we never train on the TAC 2011 data, throughout this section, we directly report our results on the TAC 2011 evaluation dataset.

Utility of Reference Mention Selection

In Table 6.7 we compare the performance of our submitted system with and without mention generation policy discussed in Section 6.11.2. We note that our submitted results to TAC were obtained using a GLOW model trained on our internal newswire dataset rather than on the TAC data and with MaxNoThres solution aggregation strategy. The baseline is to mark all the mentions in the query text which exactly match the query form, we call this policy Naive mention generation. We compared the performance of the Naive strategy and the strategy discussed in Section 6.11.2, which as the table shows improves the micro-average performance by 3 points and the B^3 F1 by 4 points.

Utility of Solution Aggregation Strategies

In Section 6.11.4 we have mentioned that given a knowledge base $KB = \{E_1, E_2, \ldots, E_{|KB|}\}$, a query

Mention Generation	Performance			
Policy	Micro-Average	B^3 Precision	B^3 Recall	B^3 F1
Naive	0.752	0.709	0.740	0.724
According to Sec 6.11.2	0.787	0.757	0.765	0.761

Table 6.7: The utility of mention selection. The Naive mention generation strategy is marking all the mention in query text which match exactly the query surface form. The method for mention generation proposed in Section 6.11.2 improves the micro-average performance by 3 points and the B^3 F1 by 4 points. We note that these results were obtained using the GLOW model trained on our internal newswire dataset rather than on the TAC data and with MaxNoThres solution aggregation strategy.

Mention Generation	Performance			
Policy	Micro-Average	B^3 Precision	B^3 Recall	B^3 F1
MaxNoThres	0.787	0.757	0.765	0.761
MaxWithThres	0.788	0.757	0.765	0.761
SumNoThres	0.794	0.763	0.773	0.768
SumWithThres	0.788	0.757	0.766	0.762

Table 6.8: Comparison of the solution generation policies. We note that these results were obtained using the GLOW model trained on our internal newswire dataset rather than on the TAC data. All approaches are competitive.

$(Q_{id}, Q_{form}, Q_{text})$ we use GLOW to generate a set of tuples $\{(m_i, t_i, r_i, l_i)\}_{1 \le i \le N}$. However, our end goal is to assign a KB single entry E^* to the query, and we have suggested four approaches for generating a single solution, namely: MaxNoThres, MaxWithThres, SumNoThres, SumWithThres. In Table 6.8 we compare these approaches and conclude that all approaches are competitive.

Ablation Feature Study The GLOW system has several groups of features: baseline, lexical naive, lexical re-weighted, and coherence[5]. In [111] we ran an ablation study on the Wikification task, assessing the strengths and the weaknesses of each feature group. We concluded that the baseline features provide a very strong baseline. Lexical features lead to state-of-the-art performance, and while adding coherence features allow to further marginally improve the performance, the key difficulty was identifying when a mention refers to out-of-Wikipedia entity In other words, the linker scores are not very reliable. One of our goals of participating in the TAC KBP entity linking competition was to see whether these statements hold true for the TAC KBP entity linking task. In the following set of experiments, we have used the models obtained in [111] for different feature groups. All of the models were trained on around 10K paragraphs from Wikipedia articles. In Table 6.9 we compare the performance of the different GLOW models using different sets of features. We note that in all the experiments, we used our mention selection strategy from Section 6.11.2 and the SumNoThres single-solution generation strategy.

There are several conclusions in place. First, both the lexical features and the coherence features have

[5] [111] has compared multiple approaches to capture coherence. In this work, we only report the best-performing approach: when disambiguating mention m, use baseline predictions for other mentions as a semantic context.

	Performance			
Features Used	Micro-Average	B^3 Precision	B^3 Recall	B^3 F1
Baseline	0.747	0.710	0.731	0.720
Baseline+Lexical				
Naive	0.784	0.749	0.764	0.756
Re-weighted	—	—	—	
All Lexical	0.786	0.752	0.766	0.759
Baseline+Global				
Coherence	0.780	0.749	0.760	0.754
Baseline+Local+Global				
All features	0.783	0.754	0.759	0.756

Table 6.9: Ranker Accuracy. Bold values indicate the best performance in each feature group. The global approaches marginally outperform the local approaches on *ranker accuracy* , while combing the approaches leads to further marginal performance improvement.

improved the performance considerably over the baseline. Second, surprisingly, both the lexical and the coherence features performed extremely competitively to one another, and combining them did not lead to further improvement. Surprisingly, the naive lexical features performed almost as well as the re-weighted lexical features, which in [111] performed significantly better. Finally, the best configuration of models trained on the 10K paragraphs from Wikipedia articles achieved macro-average of 0.786 and B^3 F1 of 0.759, while the best configuration trained on 200 newswire documents achieved achieved macro-average of 0.794 and B^3 F1 of 0.768. Which means that a system trained on a smaller amount of newswire data and blogs marginally outperformed a system trained on a large amount of Wikipedia data. We hypothesize that the majority of test document contain enough context to easily disambiguate the mentions, as long as the correct mentions have been identified (correct being indeed matching the query form) and the correct disambiguation appears in the disambiguation candidate list.

6.11.6 Conclusions

We have presented an approach for using the GLOW system for the TAC KBP entity linking challenge. Our approach was based on detecting mentions matching the query form in the query text, disambiguating them to Wikipedia using GLOW and then forming a single Wikipedia title t corresponding to the query. Finally, we have matched the assigned Wikipedia title to the KBP knowledge base using a February 2011 set of Wikipedia redirects. We noticed that although the GLOW system did not use the TAC KBP entity linking data for training or tuning, it achieved a surprisingly good performance. We noticed that matching the query form against potential mentions in the query text has a major impact on the end performance, allowing us to improve by 4 points B^3 F1 over the baseline. All reasonable strategies for reconciling potentially conflicting disambiguations for the identified mention set, such as MaxNoThres, MaxWithThres, SumNoThres, SumWithThres led to similar performance. GLOW has several feature groups. All of them performed similarly, and surprisingly a combination of multiple lexical features or lexical and coherence features together did not lead to an improvement over a single feature group. We were also surprised to discover that the GLOW system trained on 200 newswire documents outperformed the same system when trained on 10K articles from Wikipedia. Overall, the selection of the training set did not have much impact, and most of the performance gains were made through our approach for detecting mentions matching the query form in the query text and a single (either lexical or coherence) feature group. We hypothesize that the majority of test document contain enough context to easily disambiguate the mentions, as long as the correct mentions have been identified (correct being indeed matching the query form) and the correct disambiguation appears in the disambiguation candidate list.

Chapter 7

Co-reference Resolution

7.1 Introduction

Co-reference resolution is the task of grouping mentions to entities. For example, consider the text snippet in Fig. 7.1[1]. The correct output groups the mentions $\{m_1, m_2, m_5\}$ to one entity while leaving m_3 and m_4 as singletons. Resolving co-reference is fundamental for understanding natural language. For example in Fig. 7.1, to infer that Kusrk has suffered a torpedo detonation, we have to understand that $\{[\textit{vessel}]\}_{m1}$ refers to $\{[\textit{Kursk}]\}_{m2}$.

This inference is typically trivial for humans, but proves extremely challenging for state-of-the-art co-reference resolution systems. We believe that it is world knowledge that gives people the ability to understand text with such ease. A human reader can infer that since Kursk sank, it must be a vessel and vessels which suffer catastrophic torpedo detonations can sink. Moreover, some readers might *just know* that *Kursk* is a Russian submarine named after the city of Kursk, where the largest tank battle in history took place in 1943.

In this work we are using Wikipedia as a source of encyclopedic knowledge. Fig. 7.2 gives a high-level overview of our system. We are using a publicly available system for context-sensitive disambiguation to Wikipedia. We then extract attributes from the cross-linked Wikipedia pages (covered in Section 7.3.1), assign these attributes to the document mentions (Section 7.3.2) and develop knowledge-rich compatibility metric between mentions (Section 7.3.3)[2]. To maximize the utility of the injected knowledge, we develop

[1]Throughout this paper, curly brackets {} denote the extent and square brackets [] denote the head.

[2]The extracted attributes and the related resources are available for public download at `http://cogcomp.cs.illinois.edu/Data/Ace2004CorefWikiAttributes.zip`

"After the $\{[\textit{vessel}]\}_{m_1}$ suffered a catastrophic torpedo detonation, $\{[\textit{Kursk}]\}_{m_2}$ sank in the waters of $\{[\textit{Barents Sea}]\}_{m_3}$ with all hands lost. Though rescue attempts were offered by a nearby $\{\textit{Norwegian } [\textit{ship}]\}_{m_4}$, Russia declined initial rescue offers, and all 118 sailors and officers aboard $\{[\textit{Kursk}]\}_{m_5}$ perished."

Figure 7.1: Example illustrating the challenges in co-reference resolution.

```
Input: document $d$; mentions $M = \{m_1, \ldots, m_N\}$
1) For each $m_i \in M$, assign it a Wikipedia page $p_i$ in a context-sensitive way ($p_i$ may be $null$).
- If $p_i \neq null$: extract knowledge attributes from $p_i$ and assign to $m$.
- Else extract knowledge attributes directly from $m$ via noun-phrase parsing techinques [150].
3) Let $Q = \{(m_i, m_j)\}_{i \neq j}$, be the queue of mention pairs *approximately* sorted by "easy-first" [52].
4) Let $G$ be a partial clustering graph.
5) While $Q$ is not empty
- Extract a pair $p = (m_i, m_j)$ from $Q$.
- Using the knowledge attributes of $m_i$ and $m_j$ as well as the structure of $G$, classify whether $p$ is co-referent.
- Update $G$ with the classification decision.
6) Construct an end clustering from $G$.
```

Figure 7.2: High-level system architecture.

approaches for entity-level inference in Section 7.4. We conclude with experiments in Section 7.6 and discussion in Section 7.7.

The key contributions of this work are:

(1) Using Wikipedia to assign a set of knowledge attributes to mentions in a context-sensitive way.

(2) Integrating the strength of rule-based systems such as [56; 108] into a machine learning framework.

(3) A novel approach for entity-level inference.

(4) By adding word-knowledge features, we improve the performance of a state-of-the-art system of [10] by 3 MUC, 2 B^3 and 2 CEAF F1 points on the non-transcript portion of the ACE 2004 dataset.

7.2 Baseline System and Data

In this work, we are using the state-of-the-art system of [10], which relies on a pairwise scoring function pc to assign an ordered pair of mentions a probability that they are coreferential. It uses a rich set of features including: string edit distance, gender match, whether the mentions appear in the same sentence, whether the heads are synonyms in WordNet etc. The function pc is modeled using regularized averaged perceptron for a tuned number of training rounds, learning rate and margin. For the end system, we keep these parameters intact, our only modifications will be adding knowledge-rich features and adding intermediate classification layers to the training and the inference, which we will discuss in the following sections.

At inference time, given a document d and a pairwise co-reference scoring function pc, [10] generate a graph G_d according to the Best-Link decision model [99] as follows. For each mention m in document d, let B_m be the set of mentions appearing before m in d. Let a be the highest scoring antecedent: $a = \text{argmax}_{b \in B_m}(pc(b, m))$. We will add the edge (a, m) to G_d if $pc(a, m)$ predicts the pair to be co-referent with a confidence exceeding a chosen threshold, then we take the transitive closure[3].

The properties of the Best-Link inference are illustrated in Fig. 7.3. At this stage, we ask the reader to ignore the knowledge attributes at the bottom of the figure. Let us assume that the pairwise classifier labeled the mentions (m_2, m_5) co-referent because they have identical surface form; mentions (m_1, m_4) are co-referred because the heads are synonyms in WordNet. Let us assume that since m_1 and m_2 appear in the same sentence, the pairwise classifier managed to leverage the dependency parse tree to correctly co-ref the pair (m_1, m_2). The transitive closure would correctly link (m_1, m_5) despite the incorrect prediction of the pairwise classifier on (m_1, m_5), and would incorrectly link m_4 with all other mentions because of the incorrect pairwise prediction on (m_1, m_4) and despite the correct predictions on (m_2, m_4) and (m_4, m_5).

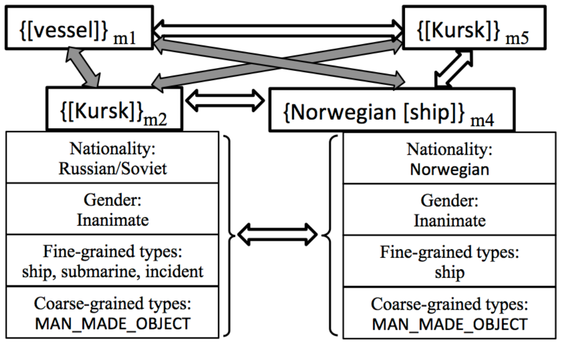

{[vessel]} m1
{[Kursk]} m5
{[Kursk]}m2
{Norwegian [ship]} m4
Nationality: Russian/Soviet
Gender: Inanimate
Fine-grained types: ship, submarine, incident
Coarse-grained types: MAN_MADE_OBJECT
Nationality: Norwegian
Gender: Inanimate
Fine-grained types: ship
Coarse-grained types: MAN_MADE_OBJECT

Figure 7.3: A sample output of a pairwise co-reference classifier. The full edges represent a co-ref prediction and the empty edges represent a non-coref prediction. A set of knowledge attributes for selected mentions is shown as well.

We use the official ACE 2004 English training data [102]. We started with the data split used in [30], which used 336 documents for training and 107 documents for testing. We note that ACE data contains both newswire text and transcripts. In this work, we are using NLP tools such as POS tagger, named entity recognizer, shallow parser, and a disambiguation to Wikipedia system to inject expressive features into a co-reference system.

Unfortunately, current state-of-the-art NLP tools do not work well on transcribed text. Therefore, we discard all the transcripts. Our criteria was simple. The ACE annotators have marked the named entities both in newswire and in the transcripts. We kept only those documents which contained named entities (according to manual ACE annotation) and at least 1/3 of the named entities started with a capital letter.

[3] We use Platt Scaling while [10] used the raw output value of pc.

After this pre-processing step, we were left with 275 out of the original 336 training documents, and 42 out of the 107 testing documents.

7.3 Wikipedia as Knowledge

In this section we describe our methodology for using Wikipedia as a knowledge resource. In Section 7.3.1 we cover the process of knowledge extraction from Wikipedia pages. We describe how to inject this knowledge into mentions in Section 7.3.2. The bottom part of Fig. 7.3 illustrates the knowledge attributes our system injects to two sample mentions at this stage. Finally, in Section 7.3.3 we describe a compatibility metric our system learns over the injected knowledge.

7.3.1 Wikipedia Knowledge Attributes

Our goal in this section is to extract from Wikipedia pages a compact and highly-accurate set of **knowledge attributes**, which nevertheless possesses discriminative power for co-reference[4]. We concentrate on three types of knowledge attributes: fine-grained semantic categories, gender information and nationality where applicable.

Each Wikipedia page is assigned a set of categories. There are over 100K categories in Wikipedia, many are extremely fine-grained and contain very few pages. The value of Wikipedia category structure for knowledge acquisition has long been noticed in several influential works, such as [134; 97] to name a few. However, while the recall of the above resources is excellent, we found their precision insufficient for our purposes. We have implemented a simple high-precision low-recall heuristic for extracting the head words of Wikipedia categories as follows.

We noticed that Wikipedia categories have a simple structure of either *<noun-phrase>* or (*<noun-phrase><relation-token><noun-phrase>*), where in the second case the category information is always on the left. Therefore, we first remove the text succeeding a set of carefully chosen relation tokens[5]. With this heuristic *"Recipients of the Gold Medal of the Royal Astronomical Society"* becomes just *"Recipients"*; *"Populated places in Africa"* becomes *"places"*; however *"Institute for Advanced Study faculty"* becomes *"Institute"* (rather than *"faculty"*). At the second step, we apply the Illinois POS tagger and keep only the tokens labeled as NNS. This step allows us to exclude singular nouns incorrectly identified as heads, such as *"Institute"* above. To further reduce the noise in category extraction, we also remove all rare category tokens which appeared in less than 100 titles ending up with 2088 fine-grained entity types. We manually map popular

[4]We justify the reasons for our choice of high-precision low-recall knowledge extraction in Section 7.3.2.

[5]The selected set was: {of, in, with, from, ",", at, who, which, for, and, by}

fine-grained categories to coarser-grained ones, more consistent with ACE entity typing. A sample of the mapping is shown in the table below:

Fine-grained	Coarse-grained
people, births, writers, inventors, ...	PER
departments, organizations, banks, ...	ORG
venues, trails, areas, buildings, ...	LOC
countries, towns, villages, ...	GPE
churches, highways, schools, ...	FACILITY

Manual inspection of the extracted category keywords has led us to believe that this heuristic achieves a higher precision at a considerable loss of recall when compared to the more sophisticated approach of [97], which correctly identifies *"faculty"* as the head of *"Institute for Advanced Study faculty"*, but incorrectly identifies *"statistical organizations"* as the head of *"Presidents of statistical organizations"* in about half the titles containing the category[6].

We assign gender to the titles using the following simple heuristic. The first paragraph of each Wikipedia article provides a very brief summary of the entity in focus. If the first paragraph of a Wikipedia page contains the pronoun "she", but not "he", the article is considered to be about a female (and vice-versa). However, when the page is assigned a non-person-related fine-grained NE type (e.g. school) and at the same time is not assigned a person-related fine-grained NE type (e.g. novelist), we mark the page as inanimate regardless of the presence of he/she pronouns. The nationality is assigned by matching the tokens in the original (unprocessed) categories of the Wikipedia page to a list of countries. We assign nationality not only to the Wikipedia titles, but also to single tokens. For each token, we track the list of titles it appears in, and if the union of the nationalities assigned to the titles it appears in is less than 7, we mark the token compatible with these nationalities. This allows us to identify Ivan Lebedev as Russian and Ronen Ben-Zohar as Israeli, even though Wikipedia may not contain pages for these specific people.

7.3.2 Injecting Knowledge Attributes

Once we have extracted the knowledge attributes of Wikipedia pages, we need to inject them into the mentions. [109] used YAGO for similar purposes, but noticed that knowledge injection is often noisy. Therefore they used YAGO only for mention pairs where one mention was an NE of type PER/LOC/ORG and the other was a common noun. This implies that all MISC NEs were discarded, and all NE-NE pairs

[6][97] analyze a set of categories S assigned to Wikipedia page p jointly, hence the same category expression can be interpreted differently, depending on S.

were discarded as well. We also note that [109] reports low utility of FrameNet-based features. In fact, when incrementally added to other features in cluster-ranking model the FrameNet-based features sometimes led to performance drops. This observation has motivated our choice of high-precision low-recall heuristic in Section 7.3.1 and will motivate us to add features conservatively when building attribute compatibility metric in Section 7.3.3.

Additionally, while [109] uses the union of all possible meanings a mention may have in Wikipedia, we deploy GLOW [111][7], a context-sensitive system for disambiguation to Wikipedia. Using context-sensitive disambiguation to Wikipedia as well as high-precision set of knowledge attributes allows us to inject the knowledge to more mention pairs when compared to [109]. Our exact heuristic for injecting knowledge attributes to mentions is as follows:

Named Entities with Wikipedia Disambiguation

If the mention head is an NE matched to a Wikipedia page p by GLOW, we import all the knowledge attributes from p. GLOW allows us to map *"E**ph**raim Sneh"* to *http://en.wikipedia.org/wiki/E**f**raim_Sneh* and to assign it the *Israeli* nationality, *male* gender, and the fine-grained entity types: {*member, politician, person, minister, alumnus, physician, general*}.

Head and Extent Keywords

If the mention head is not mapped to Wikipedia by GLOW and the head contains keywords which appear in the list of 2088 fine-grained entity types, then the rightmost such keyword is added to the list of mention knowledge attributes. If the head does not contain any entity-type keywords but the extent does, we add the rightmost such keyword of the extent. In both cases, we apply the heuristic of removing clauses starting with a select set of punctuations, prepositions and pronouns, annotating what is left with POS tagger and restricting to noun tokens only[8]. This allows us to inject knowledge to mentions unmapped to Wikipedia, such as: *"{current Cycle World publisher [Larry Little]}"*, which is assigned the attribute *publisher* but not *world* or *cycle*. Likewise, *"{[Joseph Conrad Parkhurst], who founded the motorcycle magazine Cycle World in 1962 }"*, is not assigned the attribute *magazine*, since the text following *"who"* is discarded.

7.3.3 Learning Attributes Compatibility

In the previous section we have assigned knowledge attributes to the mentions. Some of this information, such as gender and coarse-grained entity types are also modeled in the baseline system of [10]. Our goal is to build a compatibility metric on top of this redundant, yet often inconsistent information.

[7] Available at: `http://cogcomp.cs.illinois.edu/page/software_view/Wikifier`

[8] This heuristic is similar to the one we used for extracting Wikipedia category headwords and seems to be a reasonable baseline for parsing noun structures [150].

The majority of the features we are using are straightforward, such as: (1) whether the two mentions mapped to the same Wikipedia page, (2) gender agreement (both Wikipedia and dictionary-based), (3) nationality agreement (here we measure only whether the sets intersect, since mentions can have multiple nationalities in the real world), (4) coarse-grained entity type match, etc.

The only non-trivial feature is measuring compatibility between sets of fine-grained entity types, which we describe below. Let us assume that mention m_1 was assigned the set of fine-grained entity types S_1 and the mention m_2 was assigned the set of fine-grained entity types S_2. We record whether S_1 and S_2 share elements. If they do, than, in addition to the boolean feature, the list of the shared elements also appears as a list of discrete features. We do the same for the most similar and most dissimilar elements of S_1 and S_2 (along with their discretized similarity score) according to a WordNet-based similarity metric of [37]. The reason for explicitly listing the shared, the most similar and dissimilar elements is that the WordNet similarity does not always correspond to co-reference compatibility. For example, the pair $(company, rival)$ has a low similarity score according to WordNet, but characterizes co-referent mentions. On the other hand, the pair $(city, region)$ has a high WordNet similarity score, but characterizes non-coreferent mentions. We want to allow our system to "memorize" the discrepancy between the WordNet similarity and co-reference compatibility of specific pairs.

We also note that we generate a set of selected conjunctive features, most notably of fine-grained categories with NER predictions. The reason is that the pair of mentions *"(Microsoft, Google)"* are not co-referent despite the fact that they both have the *company* attribute. On the other hand *"(Microsoft, Redmond-based company)"* is a co-referent pair. To capture this difference, we generate the feature *ORG-ORG&&share_attribute* for the first pair, and *ORG-O&&share_attribute* for the second pair[9]. These features are also used in conjunction with string edit distance. Therefore, if our system sees two named entities which share the same fine-grained type but have a large string edit distance, it will label the pair as non-coref.

7.4 Learning-based Multi-Sieve Approach

State-of-the-art machine-learning co-ref systems, e.g. [10; 109] train a single model for predicting co-reference of all mention pairs. However, rule-based systems, e.g. [56; 108] characterize mention pairs by discourse structure and linguistic properties and apply rules in a prescribed order (high-precision rules first). Somewhat surprisingly, such hybrid approach of applying rules on top of structures produced by statistical tools (such as dependency parse trees) performs better than pure machine-learning approach[10].

[9]The head of *"Redmond-based company"* is *"company"*, which is not a named entity, and is marked O.
[10][108] recorded the best result on CoNLL 2011 shared task.

In this work, we attempt to integrate the strength of linguistically motivated rule-based systems with the robustness of a machine learning approach. We started with a hypothesis that different types of mention pairs may require a different co-ref model. For example, consider the text below:

> *Queen Rania of Jordan , Egypt's [Suzanne Mubarak]$_{m_1}$ and others were using their charisma and influence to campaign for equality of the sexes. [Mubarak]$_{m_2}$, wife of Egyptian President [Hosni Mubarak]$_{m_3}$, and one of the conference organizers, said they must find ways to ...*

There is a subtle difference between mention pairs (m_1, m_2) and (m_2, m_3). One of the differences is purely structural. The first pair appears in different sentences, while the second pair – in the same sentence. It turns out that string edit distance feature between two named entities has different "semantics" depending on whether the two mentions appear in the same sentence. The reason is that to avoid redundancy, humans refer to the same entity differently within the sentence, preferring titles, nicknames and pronouns. Therefore, when a similar-looking named entities appear in the same sentence, they are actually likely to refer to different entities. On the other hand, in the sentence *"Reggie Jackson, nicknamed Mr. October ..."* we have to rely heavily on sentence structure rather than string edit distance to make the correct co-ref prediction.

Layer	Trained on All Data	Layer-specific Training
AllSentencePairs	61.37	67.46
ClosestNonProDiffSent	60.71	63.33
NonProSameSentence	62.97	63.80
NerMentionsDiffSent	86.44	87.12
SameSentenceOneNer	64.10	68.88
Adjacent	71.00	78.80
SameSenBothNer	75.30	73.75
Nested	76.11	79.00

Table 7.1: F1 performance on co-referent mention pairs by inference layer when trained with all data versus level-specific data only.

Our second intuition is that "easy-first" inference is necessary to effectively leverage knowledge. For example, in Figure 7.3, our goal is to link *vessel* to *Kursk* and assign it the *Russian/Soviet* nationality prior to applying the pairwise co-ref classifier on (*vessel, Norwegian ship*). Therefore, our goal is to apply the pairwise classifier on pairs in *prescribed order* and to propagate the knowledge across mentions. The ordering should be such that (a) maximum amount of information is injected at early stages (b) the precision at the early stages is as high as possible [108]. Hence, we divide the mention pairs as follows:

Nested: are pairs such as "$\{\{[city]_{m_1}\}$ *of* $[Jerusalem]_{m_2}\}$" where the extent of one of the mentions contains the extent of the other. For some mentions, the extent is the entire clause, so we also added a requirement that mention heads are at most 7 tokens apart. Intuitively, it is the easiest case of co-reference. There are 5,804 training samples and 992 testing samples, out of which 208 are co-referent.

SameSenBothNer: are pairs of named entities which appear in the same sentence. We already saw an example for this case involving $[\text{Mubarak}]_{m2}$ and $[\text{Hosni Mubarak}]_{m3}$. There are 13,041 training samples and 1,746 testing samples, out of which 86 are co-referent.

Adjacent: are pairs of mentions which appear closest to each other on the dependency tree. We note that most of the nested pairs are also adjacent. There are training 5,872 samples and 895 testing samples, out of which 219 are co-referent.

SameSentenceOneNer: are pairs which appear in the same sentence and exactly one of the mentions is a named entity, and the other is not a pronoun. Typical pairs are *"Israel-country"*, as opposed to *"Bill Clinton - reporter"*. This type of pairs is fairly difficult, but our hope is to use encyclopedic knowledge to boost the performance. There are 15,715 training samples and 2,635 testing samples, out of which 207 are co-referent.

NerMentionsDiffSent: are pairs of mentions in different sentences, both of which are named entities. There are 189,807 training samples and 24,342 testing samples, out of which 1,628 are co-referent.

NonProSameSentence: are pairs in the same sentence, where both mentions are non-pronouns. This level includes all the pairs in the SameSentenceOneNer level. Typical pairs are *"city-capital"* and *"reporter-celebrity". There are 33,895 training samples and 5,393 testing samples, out of which 336 are co-referent.*

ClosestNonProDiffSent: are pairs of mentions in different sentences with no other mentions between the two. 3,707 training samples and 488 testing samples, out of which 38 are co-referent.

AllSentencePairs: All mention pairs within same sentence. There are 49,953 training samples and 7,809 testing samples, out of which 846 are co-referent.

FinalLayer: The set of mention pairs classified by the baseline system. 525,398 training samples and 85,358 testing samples, out of which 1,387 are co-referent.

In Table 7.1 we compare the performance of the model layer by layer in two scenarios. First, we train with the entire 525,398 training samples, and then we train on whatever training data is available for the specific layer[11]. We were surprised to see that the F1 on the nested mentions, when trained on the 5,804 layer-specific samples improves to 79.00 versus 76.11 when trained on the 525,398 final layer samples.

There are several things to note when interpreting the results in Tab 7.1. First, the sheer ratio of positive to negative samples fluctuates drastically. For example, 208 out of the 992 testing samples at the nested layer are positive, while there are only 86 positive samples out of 1,746 testing samples in the SameSenBothNer layer. It seems unreasonable to use the same model for inference at both layers. Second, the data for intermediate layers is not always a subset of the final layer. The reason is that final layer extracts a positive instance only for the closest co-referent mentions, while layers such as AllSentencePairs extract samples for all co-referent pairs which appear in the same sentence. Third, while our division to layers may resemble witchcraft, it is motivated by the intuition that mentions appearing close to one another are easier instances of co-ref as well as linguistic insights of [108].

7.5 Entity-Level Inference

In this section we describe our approach for building entity-level features. Let $\{C_1, C_2, \dots C_N\}$ be the set of layer-specific classifiers. In our case, C_1 is the nested mention pairs classifier, C_2 is the SameSenBothNer classifier, and C_9 is the final layer classifier. We design layer-based features so that the subsequent layers "see" the decisions of the previous layers and use entity-level features based on the intermediate clustering. However, unlike [108], we allow the subsequent layers to change the decisions made by the lower layers (since additional information becomes available).

Let $R_i(m)$ be the set of all mentions which, when paired with the mention m form valid sample pairs for layer i. E.g. in our running example of Figure 7.1, $R_2([Kursk]_{m_2}) = \{[\textit{Barents Sea}]_{m_3}\}$, since both m_1 and m_2 are NEs and appear in the same sentence. Let $R_i^+(m)$ be the set of all mentions which were labeled as co-referent to the mention m by the classifier C_i (including m, which is co-referent to itself). We define $R_i^-(m)$ similarly. We denote the union of mentions co-refed to m during inference up to layer

[11] We report *pairwise* performance on mention pairs because it is the more natural metric for the intermediate layers. We report only performance on co-referent pairs, because for many layers, such as the final layer, 99% of the mention pairs are non-coref, hence a baseline of labeling all samples as non-coref would result in 99% accuracy. We are interested in a more challenging baseline, the co-referent pairs.

i as $E_i^+(m) = \cup_{j=1}^{i-1} R_j^+(m)$. Similarly, $E_i^-(m) = \cup_{j=1}^{i-1} R_j^-(m)$. Using these definitions we can introduce entity-level prediction features which allow subsequent layers to use information aggregated from previous layers.

Intermediate Clustering Features (IC)

$$IC_i^R(m_j, m_k) = \begin{cases} -1 & m_j \in R_{i-1}^-(m_k) \\ +1 & m_j \in R_{i-1}^+(m_k) \\ 0 & \text{Otherwise} \end{cases}$$

$$IC_i^E(m_j, m_k) = \begin{cases} -1 & m_j \in E_{i-1}^-(m_k) \\ +1 & m_j \in E_{i-1}^+(m_k) \\ 0 & \text{Otherwise} \end{cases}$$

IC_i^R stores the pairwise prediction history, thus when classifying a pair (m_j, m_k) at layer i, a classifier can see the predictions of all the previous layers applicable on that pair. IC_i^E stores the transitive closures of the layer-specific predictions. We note that both IC_i^R and IC_i^E can have the values +1 and -1 active at the same time if intermediate layer classifiers generated conflicting predictions. However, a classifier at level i will use as features both $IC_1^R, \ldots IC_{i-1}^R$ and $IC_1^E, \ldots IC_{i-1}^E$, thus it will know the lowest layer at which the conflicting evidence occurs. The classifier at level i also uses set identity, set containment, set overlap and other set comparison features between $E_{i-1}^{+/-}(m_j)$ and $E_{i-1}^{+/-}(m_k)$. We check whether the sets have symmetric difference, whether the size of the intersection between the two sets is at least half the size of the smallest set etc. We also generate subtypes of set comparison features when restricting the elements to NE-mentions and non-pronominal mentions (e.g "what percent of named entities do the sets have in common?").

Surface Form Compatibility (SFC)

The intermediate clustering features do not allow us to generalize predictions from pairs of mentions to pairs of surface strings. For example, if we have three mentions: $\{[\textit{vessel}]_{m_1}, [\textit{Kursk}]_{m_2}, [\textit{Kursk}]_{m_5}\}$, then the prediction on the pair (m_1, m_2) will not be used for the prediction on the pair (m_1, m_5), even though in both pairs we are asking whether *Kursk* can be referred to as *vessel*. The surface form compatibility features mirror the intermediate clustering features, but relax mention IDs and replace them by surface forms. Similarly to intermediate clustering features, both +1 and -1 values can be active at the same time. We also generate subtypes of set-comparison features for NE-mentions and optionally stemmed non-pronominal mentions.

7.6 Experiments and Results

	(B)aseline	(B)+Knowledge	(B)+Predictions	(B)+Knowledge +Predictions
FinalLevel	66.58	69.08	68.77	**70.43**
AllSentencePairs	67.46	71.79	69.59	**73.50**
ClosestNonProDiffSent	63.33	65.62	65.57	**70.76**
NonProSameSentence	63.80	69.62	67.03	**71.11**
NerMentionsDiffSent	87.12	88.23	88.68	**89.07**
SameSentenceOneNer	68.88	70.58	67.89	**73.17**
Adjacent	78.80	81.32	80.00	**81.79**
SameSenBothNer	73.75	80.50	77.21	**80.98**
Nested	79.00	83.59	80.65	**83.37**

Table 7.2: Utility of knowledge and prediction features (F1 on co-referent mention pairs) by inference layers. Both knowledge and entity-based features significantly and independently improve the performance for all layers. The goal of entity-based features is to propagate knowledge effectively, thus it is encouraging that the combination of entity-based and knowledge features performed significantly better than any of the approaches individually at the final layer.

7.6.1 Ablation Study

In Table 7.2 we report the pairwise F1 scores on co-referent mention pairs broken down by layer and using different components. This allows us to see, for example, that adding only the knowledge attributes improved the performance at *NonProSameSentence* layer from 63.80 to 69.62. We have ordered the layers according to our initial intuition of "easy first". We were surprised to see that co-ref resolution for named entities in the same sentence was harder than cross-sentence (73.75 vs. 87.12 baseline F1). We were also surprised to see that resolving all mention pairs within sentence when including pronouns was easier than resolving pairs where both mentions were non-pronouns (67.46 vs. 63.80 baseline F1).

7.6.2 End system performance

Recall that the Best-Link algorithm applies transitive closure on the graph generated by thresholding the pairwise co-reference scoring function pc. The lower the threshold on the positive prediction, the lower is the precision and the higher is the recall. In Fig. 7.4 we compare the end clustering quality across a variety of thresholds and for various system flavors using three metrics: MUC [157], B^3 [2] and CEAF [84][12]. The purpose of this comparison is to see the impact of the knowledge and the prediction features on the final output and to see whether the performance gains are due to (mis-)tuning of one of the systems or are they consistent across a variety of thresholds.

[12]In the interest of space, we refer the reader to the literature for details about the different metrics.

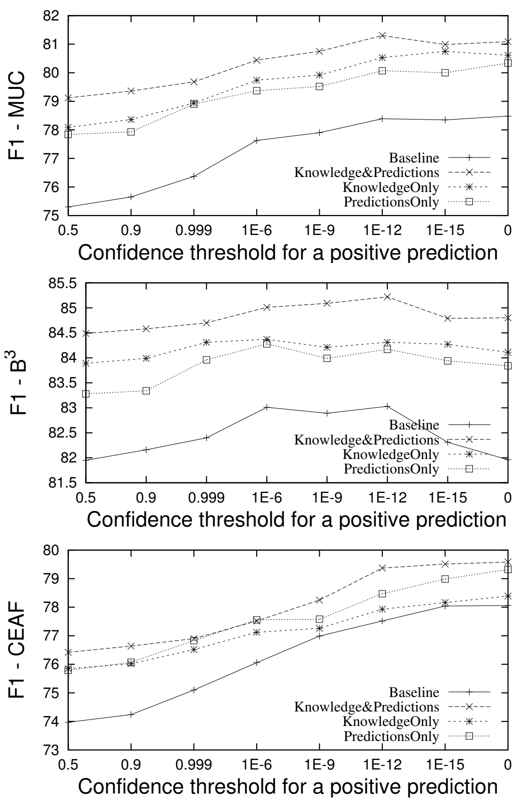


Figure 7.4: End performance for various systems.

The end performance of the baseline system on our training/testing split peaks at around 78.39 MUC, 83.03 B^3 and 77.52 CEAF, which is higher (e.g. 3 B^3 F1 points) than the originally reported result on the entire dataset (which includes the transcripts). This is expected, since well-formed text is easier to process than transcripts. We note that our baseline is a state-of-the art system which recorded the highest B^3 and BLANC scores at CoNLL 2011 shared task and took the third place overall. Fig. 7.4 shows a minimum improvement of 3 MUC, 2 B^3 and 1.25 CEAF F1 points across all thresholds when comparing the baseline to our end system. Surprisingly, the knowledge features outperformed prediction features on pairwise, MUC and B^3 metrics, but not on the CEAF metric. This shows that pairwise performance is not always indicative of cluster-level performance for all metrics.

7.7 Conclusions and Related Work

To illustrate the strengths of our approach, let us consider the following text:

Another terminal was made available in $\{[\textit{Jiangxi}]_{m_1}\}$, *an* $\{\textit{inland } [\textit{province}]_{m_2}\}$.

. . . The previous situation whereby large amount of goods for $\{\textit{Jiangxi } [\textit{province}]_{m_3}\}$ *had to be re-shipped through Guangzhou and Shanghai will be changed completely.*

The baseline system assigns each mention to a separate cluster. The pairs (m_1, m_2) and (m_1, m_3) are misclassified because the baseline classifier does not know that *Jiangxi* is a *province* and the preposition *an* before m_2 is interpreted to mean it is a previously unmentioned entity. The pair (m_2, m_3) is misclassified because identical heads have different modifiers, as in *(big province, small province)*. Our end system first co-refs (m_1, m_2) at the *AllSameSentence* layer due to the knowledge features, and then co-refs (m_1, m_3) at the final layer due to surface form compatibility features indicating that *province* was observed to refer to *Jiangxi* in the document. The transitivity of Best-Link takes care of (m_2, m_3).

However, our approach has multiple limitations. Entity-level features currently do not propagate knowledge attributes directly, but through aggregating pairwise predictions at knowledge-infused intermediate layers. We attempted to propagate the knowledge explicitly across partial clustering without success due to conflicting predictions and error propagation across layers. We rely on gold mention boundaries and exhaustive gold co-reference annotation. This prevented us from applying our approach to the Ontonotes dataset where singleton clusters and co-referent nested mentions are removed. Therefore gold annotation for training several layers of our scheme is missing (e.g. nested mentions). Another limitation is our somewhat preliminary division to layers. [158] have experimented with approaches for automatic decomposition of data to subclasses and learning multiple models to improve data separability. We hope that similar approach would be useful for co-reference resolution. Ideally, we want to make "simple decisions" first, similarly to what was done in [52] for dependency parsing, and model clustering as a structured problem, similarly to [66; 160]. However, our experience with multi-sieve approach with classifiers suggests that a single model would not perform well for both lower layers with little entity-level information and higher layers with a lot of entity-level features. Addressing the aforementioned challenges is a subject for future work.

There has been an increasing interest in knowledge-rich co-reference resolution [104; 57; 109]. We use Wikipedia differently from [104] who focus on using WikiRelate, a Wikipedia-based relatedness metric [133]. [109] used the union of all possible interpretations a mention may have in YAGO, which means that *Michael Jordan* could be co-refed both to a *scientist* and *basketball player* in the same document. Additionally, [109] use exact word matching, relying on YAGO's ability to extract a comprehensive set of facts offline[13]. We are the first to use context-sensitive disambiguation to Wikipedia, which received a lot of

[13]YAGO uses WordNet to expand its set of facts. For example, if *Martha Stewart* is assigned the meaning *personality* from category head words analysis, YAGO adds the meaning *celebrity* because *personality* is a direct hyponym of *celebrity* in WordNet. However, this

attention recently [15; 29; 90; 93; 111]. We extract context-sensitive, high-precision knowledge attributes form Wikipedia pages and apply (among other features) WordNet similarity metric on pairs of knowledge attributes to determine attribute compatibility.

We have integrated the strengths of rule-based systems such as [56; 108] into a multi-sieve machine learning framework. We show that training layer-specific models significantly increases the performance on most intermediate layers.

We develop a novel approach for entity-level inference. Unlike [109] who construct entities left-to-right, and similarly to [108] we resolve easy instances of co-ref to reduce error propagation in entity-level features. Unlike [108], we allow later stages of inference to change the decisions made at lower stages as additional entity-level evidence becomes available.

By adding word-knowledge features and refining the inference, we improve the performance of a state-of-the-art system of [10] by 3 MUC, 2 B^3 and 2 CEAF F1 points on the non-transcript portion of the ACE 2004 dataset.

is done offline in a context-insensitive way, which is inherently limited.

Chapter 8

Non-encyclopedic Knowledge in NLP

Over the last few years, two of the main research directions in machine learning of natural language processing have been the study of semi-supervised learning algorithms as a way to train classifiers when the labeled data is scarce, and the study of ways to exploit knowledge and global information in structured learning tasks. In this paper, we suggest to incorporate domain knowledge in semi-supervised learning algorithms. We use *constraints* as a general framework to represent common-sense knowledge and develop a novel learning protocol which unifies and can exploit several kinds of constraints. The experimental results presented in the information extraction domain exhibit that applying constraints helps the model to generate better feedback during learning, and hence the framework allows for high performance learning with significantly less training data than was possible before on these tasks.

8.1 Introduction

Natural Language Processing (NLP) systems typically require large amounts of knowledge to achieve good performance. Acquiring labeled data is a difficult and expensive task. Therefore, an increasing attention has been recently given to semi-supervised learning, where large amounts of unlabeled data are used to improve the models learned from a small training set [27; 138]. The hope is that semi-supervised or even unsupervised approaches, when given enough knowledge about the *structure* of the problem, will be competitive with the supervised models trained on large training sets. However, in the general case, semi-supervised approaches give mixed results, and sometimes even degrade the model performance [101]. In many cases, improving semi-supervised models was done by *seeding* these models with domain information taken from dictionaries or ontology [24; 27; 55; 138].

On the other hand, in the supervised setting, it has been shown that incorporating domain and problem specific structured information can result in substantial improvement in the supervised setting [142; 120].

This paper proposes a novel constraints-based learning protocol for guiding semi-supervised learning. We develop a formalism for constraints-based learning that unifies several kinds of constraints: unary, dictionary based, constraints and n-ary constraints, which encode structural information and interdependencies among possible labels. We also extend the notion of constraint violation to account for multiple levels of violating a constraint. Our protocol can be used in the presence of any learning model, including those that acquire additional statistical constraints from observed data while learning. In the experimental part of this paper we use HMMs as the underlying model, and exhibit significant reduction in the number of training examples required in two information extraction problems.

As is often the case in semi-supervised learning, the algorithm can be viewed as a process that *labels* un-labeled examples, counting on an intermediate learned model. Our algorithm pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label, along with the current learned model, unlabeled examples. Given a small amount of labeled data and a large unlabeled pool, our framework is as follows:

1) Initialize the model based on labeled data.

2) Use constraints *and the learned model to label the instances in the pool.*

3) Update the model by newly labeled data. Go to 2.

This way, we can generate *better* "training" examples during the semi-supervised learning process. The core of our approach, (2), is described in Section 8.5. The task is described in Section 8.3 and the experimental study in Section 8.6, where we show that the improvement on the training examples via the constraints indeed boosts the learned model and the proposed method significantly outperforms the traditional semi-supervised framework.

8.2 Related Work

The idea of injecting common-sense knowledge into semi-supervised learning is not new, while ideas on using knowledge as constraints in the supervised domain are more recent, but have shown a lot of promise. Several previously suggested approaches are compared with our framework below.

In the semi-supervised domain there are two main approaches. One is using the prior knowledge to accurately tailor the generative model so that it captures the domain structure. For example, [54] proposes *Diagonal Transition Models* for sequential labeling tasks where neighboring words tend to have the same labels. This is done by constraining the HMM transition matrix, an approach that was used also for other models, such as CRF. However [120] showed that this approach is not expressive enough to support more expressive, non-sequential constraints.

A second approach has been to use a small high-accuracy set of labeled tokens as a way to seed and

(a) [*AUTHOR* Lars Ole Andersen.] [*TITLE* Program analysis and specialization for the C programming language .] [*TECH-REPORT* PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen ,] [*DATE* May 1994 .]

(b) [*AUTHOR* Lars Ole Andersen . Program analysis and] [*TITLE* specialization for the] [*EDITOR* C] [*BOOKTITLE* Programming language] [*TECH-REPORT* . PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen , May] [*DATE* 1994 .]

Figure 8.1: Error analysis of the HMM model. The correct assignment was shown in (a). While the predicted label assignment (b) is generally coherent, some obvious constraints are violated. Most obviously, punctuation marks are ignored as cues for state transitions.

bootstrap the semi-supervised learning. This was used, for example, by [138; 27] in information extraction, and by [122] in POS tagging. [55] extends the dictionary-based approach to sequential labeling tasks by propagating the information given in the seeds with contextual word similarity. [24] a conceptually similar approach, with a large named-entity dictionary, where a similarity between the candidate named-entity and its matching prototype in the dictionary was encoded as a feature in a supervised classifier.

In our framework, dictionary lookup approaches can be seen as unary constraints on the output states. We extend these kinds of constraints and allow for more general, n-ary constraints.

In the supervised learning setting it has been established that incorporating global information can significantly improve performance on several NLP tasks, including information extraction and semantic role labeling. [149; 142; 120]. Our formalism is most related to this last work. But, we develop a semi-supervised learning based on this formalism. We also make use of *soft* constraints and, furthermore, extend the notion of soft constraints to account for multiple levels of constraints' violation. Conceptually, although not technically, the most related work to ours is [130] that, in a somewhat ad-hoc manner uses soft constraints to guide an unsupervised model that was crafted for mention tracking. To the best of our knowledge, we are the first to suggest a general semi-supervised protocol that is driven by soft constraints.

We propose learning with constraints - a framework that combines the approaches described above in a unified and intuitive way.

8.3 Tasks, Examples and Datasets

In section 8.4 we will develop a general framework for learning and inference with constraints. However, it is useful to illustrate the ideas on concrete problems. Therefore, in this section, we give a brief introduction to the two domains for which we have implemented our algorithms. Both domain pertain to the Information Extraction problems, the task of identifying and extracting multiple fields from a given text. Since the fields are typically related and interdependent, this kind of applications provide a good text case for an

1) Each field must be a consecutive list of words, and can appear at most once in a citation.
2) State transitions must occur on punctuation marks.
3) The citation can only start with author or editor.
4) The words *pp. pages* correspond to $PAGE$.
5) Four digit numbers starting with 20xx and 19xx correspond to $DATE$.
6) Quotations can appear only in titles.
7) The words *'note, submitted, appear'* correspond to in the field $NOTE$.
8) The words *'CA, Australia, NY'* correspond to $LOCATION$.
9) The words *'tech, technical'* correspond to $TECH\_REPORT$.
10) The words *'proc, journal, proceedings, ACM'* correspond to $JOURNAL$ or $BOOKTITLE$.
11) The words *'ed, editors'* correspond to $EDITOR$.

Table 8.1: The list of constraints for extracting fields from citations. Constraints 1 and 2 are n-ary (or global) and are difficult to inject into traditional models. While all the constraints hold for the vast majority of the data, they are violated by at least some correct labeled assignments.

approach like ours.[1]

The first task is to identify fields from citations. The data originally included 500 labeled references, and was later extended with 5,000 unannotated citations collected from papers found on the Internet. Given a citation, the task is to extract the fields that appear in the given reference. One example is in Figure 8.1. There are 13 possible fields including author, title, date, journal, location, etc.

To gain an insight to how the constraints can guide semi-supervised learning, assume that the sentence shown in Figure 8.1 appears in the unlabeled data pool. Part (a) of the figure shows the correct labeled assignment and part (b) shows the labeled assignment of an HMM trained on 30 labeled. However, if we apply the constraint that state transition can occur only on punctuation marks, the same HMM model parameters will result in the correct labeling (a). Therefore, by adding the improved labeled assignement we can generate better trainig samples during semisupervised learning.

In fact, the punctuation marks are only some of the constraints that can be applied to this problem. The set of constraints we used in our experiments appears in Table 8.1. Note that some of the constraints are non-local and are very intuitive for people, yet it is very difficult to inject this knowledge into most probabilistic models.

The second problem we consider is extracting fields from advertisements. The dataset consists of 8,767 advertisements for apartment rentals in the San Francisco Bay Area downloaded in June 2004 from the Craigslist website. 302 of the ads have been labeled with 12 fields, including *size, rent, neighborhood, features*, and so on. The data was preprocessed by using regular expressions for phone numbers, email addresses and URLs. The list of the constraints for this domain is given in Table 8.2, and was largely imported from

[1]The data for both problems is available at: http://www.stanford.edu/ grenager/data/unsupie.tgz

the list of seed words used in [55]. We had to slightly modify the set of seedwords due to difference in preprocessing.

1) State transitions can occur only on punctuation marks or the newline symbol.
2) Each field must be at least 3 words long.
3) The words *laundry, kitchen, parking* correspond to $FEATURES$.
4) The words *sq ft, bdrm* correspond to $SIZE$.
5) The word *$ *MONEY** corresponds to $RENT$.
6) The words *close, near, shopping* correspond to $NEIGHBORHOOD$.
7) The words *laundry kitchen, parking* correspond to $FEATURES$.
8) The (normalized) words *phone, email* correspond to $CONTACT$.
9) The words *immediately, begin, cheaper* correspond to $AVAILABLE$.
10) The words *roommates, respectful, drama* correspond to $ROOMMATES$.
11) The words *smoking, dogs, cats* correspond to $RESTRICTIONS$.
12) The word **http*, image, link* corresponds to $PHOTOS$.
13) The words *address, carlmont, st, cross* correspond to $ADDRESS$.
14) The words *utilities, pays, electricity* correspond to $UTILITIES$.

Table 8.2: The list of constraints for extracting fields from advertisements. Constraints 1 and 2 are n-ary (or global) and are difficult to inject into traditional models. While all the constraints hold for the vast majority of the data, they are violated by at least some correct labeled assignments.

8.4 Notation and Definitions

Consider a structured classification problem, where given an assignment $x \in X^N$ to a set of input variables, $X = X_1, \ldots, X_N$, the task is to find the best assignment $y \in Y^M$ to the output variables $Y = Y_1, \ldots, Y_M$. We denote $\mathcal{X} = 2^X$ to be the space of the possible inputs and $\mathcal{Y} = 2^Y$ to be the set of possible outputs.

We define a structured output classifier as a function $h : \mathcal{X} \to \mathcal{Y}$ that uses a global scoring function $f : \mathcal{X} \times \mathcal{Y} \to \Re$ to assign scores to each possible example/label pair. Given an input x, a desired function f will assign the highest score to the correct output y among all the possible outputs. The global scoring function is often decomposed as a weighted sum of feature functions, $f(x, y) = \sum_{i=1}^{M} \lambda_i f_i(x, y) = \boldsymbol{\lambda} \cdot F(x, y)$. This decomposition applies both to discriminative linear classifiers and to probabilistic models, such as CRFs and HMMs, in which case the linear sum corresponds to log likelyhood assigned to the input/output pair by the model. Usually the feature functions $f_i(x, y)$ are local to allow model tractability. Local feature function can capture some context for each input or output variable, yet this context is usually very limited to allow dynamic programming decoding during inference.

Now, consider a scenario where we have a set of constraints $C_1, \ldots, C_K$. We define a constraint $C : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ as a function that indicates whether the input/output sequence violates some desired properties of the output sequence. When the constraints are hard, the solution is given by

$$\arg\max_{y \in 1_C} \boldsymbol{\lambda} \cdot F(x, y)$$

Where 1_C is a subset of $\mathcal{X} \times \mathcal{Y}$ for which all C_i assign the value 1.

When the constraints are soft, we want to incur some penalty for their violation. Moreover, we want to incorporate into our cost function a measure for the *amount* of violation incurred to C by (x,y). A generic way to capture this intuition is to introduce a distance function $d(y, \mathcal{X}_{C(x)})$ between the space of outputs that respect the constraint,$\mathcal{X}_{C_i(x)}$, and the given output sequence y. One possible way to implement this distance function is by minimal Hamming distance to a sequence that respects the constraint C $d(y, \nu_{C(x)}) = \min_{(y':C(x,y')=1)} H(y, y')$. If the penalty for violating the soft constraint C_i is ρ_i, we write the score function as:

$$\arg\max_{y} \boldsymbol{\lambda} \cdot F(x, y) - \sum_{i=1}^{K} \rho_i d(y, \mathcal{X}_{C_i(x)}) \tag{8.1}$$

We call the distance $d(y, \nu_{C(x)})$ the *valuation* of the constraint C on (x, y). We will justify the choose of Hamming distance and the implementation in Section 8.5.3.

8.5 Learning and Inference with Constraints

In this section we present a new algorithm for guiding semi-supervised learning with constraints. However, it is useful to discuss first the inference procedure and the supervised training in presence of constraints. In both the supervised and the semi-supervised setting, the global cost function is given by (8.1). During the learning procedure, we will not change ρ, and $d(y, \mathcal{X}_{C_i(x)})$ is also fixed. In all cases the by "training" we refer to the training of the parameter version $\boldsymbol{\lambda}$. While our formulation allows us to train also the coefficient of the constraints valuation, we choose not to do it, since we view this as a way to bias (or enforce) the prior knowledge into the learned model, rather than allowing the data to brush it away.

In the inference stage the task is to find an output y that maximizes the cost function. When the feature functions and the constraints are local, this problem can be solved efficiently by the Viterbi algorithm. However, we want the constraints to be more expressive to guild the model and the Viterbi algorithm cannot capture the global information. Usually in this case, search techniques are often used [63; 33; 69]. More discussion can be seen in [149]. We use the popular beamsearch decoding algorithm.

8.5.1 Supervised Learning: A Review

In supervised learning with constraints two training algorithms have been proposed. One is *learning and inference (or L+I)*, where during training, the constraints are ignored. In this case, with discriminative training the weight vector $\boldsymbol{\lambda}$ is updated when maximizing $\boldsymbol{\lambda} \cdot F(x, y)$ does not yield the correct solution and in the probabilistic framework, $\boldsymbol{\lambda}$ corresponds to maximum likelyhood estimator learned by partial counting. The constraints are applied only at the inference stage (during testing), once the model is fully learned.

The second approach is *inference based learning (or IBT)* where the weights $\boldsymbol{\lambda}$ are updated only when the output sequence y which maximizes the equation 8.1 does not yield the correct solution. Basically, both algorithms can be viewed as discriminative training methods, with *IBT* being a more conservative version of *L+I*. It has been shown that *IBT* outperforms *L+I* when the local features are unable to capture the domain structure and large amounts of training data is available [106]. Reranking is another way to inject the global information [26; 142]. In this framework, the model was learned locally but then multiple solutions were chosen. Then, the global information is used to train a model to select the best solution. In this paper we used the *L+I* algorithm for the supervised setting.

8.5.2 Semi-Supervised Learning

The semi-supervised learning with constraints is done by an EM-like procedure. We initialize the model with supervised learning on a small labeled set. Given an unlabeled set: $U = \{x^{U_i}\}$, in the estimation step, the traditional EM algorithm assigns a distribution over labeled assignments $\mathcal{X}$ for each $x \in U$, and in the maximization step, the set of model parameters is learned from the distribution which was assigned in the estimation step.

However, in the presence of constraints, assigning the complete distributions in the estimation step is unfeasible since the constraints reshape the density distribution in arbitrary way. Moreover, in the general case, we do not have a generative model. Rather than assigning the complete distribution over the label assignments for each $x \in U$, we truncate the distribution to the top K assignments as returned by beamsearch. Given a set of K top assignments $y^1, \ldots, y^K$, we approximate the estimation step by assigning uniform probability to the top K candidates, and zero probability to all the rest of the output sequences. We will refer this algorithm as *top-K hard EM*.

Another way to look at what we do is the self-training perspective. Similarly to self-training, we use the current model to generate new training examples pooled from the unlabeled set. However there are two important differences. One is that in self-training, once an unlabeled sample was labeled, it is never labeled again. In our case all the samples are relabeled each iteration. The second difference is that each unlabeled

```
Input:
    $Cycles$: number of cycles of learning.
    $Tr = (x^{Tr}, y^{Tr})$: labeled training set.
    $U = \{x^U\}$: unlabeled dataset
    $F$: set of feature functions.
    $\{\rho_i\}$: set of penalties.
    $\{C_i\}$: set of constraints.
    Top-K-Inference: returns top-K labeled assignments given the cost function.
SemiSupervisedTrain:
    Initialize $\boldsymbol{\lambda} = learn(Tr, \{f_i\})$.
    For $Cycles$ iterations do:
        $T = \phi$
        For each $x \in U$
            Let $pool$ be the top $K$ outputs returned by
              Top-K-Inference$(x, \boldsymbol{\lambda}, F, \{C_i\}, \{\rho_i\})$.
            For each $y \in pool$.
                $T = T + \{(x, y)\}$
        $\boldsymbol{\lambda} = \gamma\boldsymbol{\lambda} + (1 - \gamma)learn(T, \{f_i\})$
```

Figure 8.2: Semi-supervised learning with local score functions and constraints.

example generates K labeled instances. When we do one iteration of *top-1 hard EM*, this is equivalent to self training, where all the unlabeled samples are added to the labeled pool.

There are some possible benefits from using K samples. One is that we effectively increase the training set by a factor of K (albeit by somewhat noisy examples). In case we train a probabilistic model, so the additional noise due to some incorrect components in the predicted model only means that some of the probability estimates are a bit off. The second is that selection of *top-K* labellings allows to take into account that constraint resolution may not be deterministic. Consider for example, the output 11101000 with the constraint that it should belong to the language 1^*0^*. If the two possible corrections: 11111000 and 11100000 are scored similarly by $\boldsymbol{\lambda} \cdot F(x, y)$ adding only one of the corrections to the labeled pool is undesirable.

Our training algorithm is summarized in figure 8.2. Recall that the key claim of this paper is the semi-supervised learning procedure will be improved by constraints. Note that we can replace the beamsearch procedure(with constraints) by the Viterbi algorithm, and then algorithm restores to hard EM.

One thing about the algorithm should be clarified. It is known that traditional semi-supervised training can deteriorate the performance. [101] has suggested to balance the weight of the contribution of labeled and unlabeled data to the parameters. The intuition is that when iteratively estimating the parameters with EM, we do not want the parameters to drift too far from the supervised model. In this spirit, we introduced a balancing parameter γ, that controls the convex combination of models induced by the labeled and the unlabeled data.

8.5.3 Justification and Implementation of the Inference Procedure

Recall that our cost function is given by

$$\arg\max_{y} \boldsymbol{\lambda} \cdot F(x,y) - \sum_{i=1}^{K} \rho_i d(y, \mathcal{X}_{C_i(x)})$$

We want to justify both the use of soft constraints ρ_i and the usage of the Hamming distance for $d(y, \mathcal{X}_{C_i(x)})$. Consider the task of extracting fields from advertisements. The constraint that the state transitions can only occur on punctuation marks or on newlines holds for the large majority of the gold-labeled instances, but is sometimes violated. Surprisingly, in this case, using hard constraints still improved the performance. However, it is clear that using soft constraints is a superior option. Empirically, in supervised training with 100 labeled examples on the advertisement domain, the plain HMM gave 76.3% accuracy. When the constraint violation penalty ρ was *infinity*, the accuracy improved to 78.7%, but when the penalty was set to $-log(0.1)$[2], the accuracy of the model jumped to 80.6%.

Our choice for $d(y, \nu_{C(x)})$ to be a Hamming distance is less obvious. First of all, we claim that setting $d(y, \nu_{C(x)})$ to be binary, is suboptimal. For example, consider the advertisements domain again, where we have a single constraint: state transitions must be on punctuation marks or newlines. While vast majority of the instances satisfy the constraint, some instance violate it. Now, the gold labeling is guaranteed to violate the constraint at least once. Therefore, the binary distance will be set to 1 for that instance, loosing the ability to discriminate constraint violations in other locations. This will affect both the inference stage and the semi-supervised learning.

Approximating the Hamming distance can be a computationally hard problem. In practice, we approximate it by sequentially decoding the labels sequence with beamsearch, and incrementing the violation count for each label that violated the constraint with respect to the prefix of the current labeled assignment in the beam.

8.6 Experiments and Results

In this section, we present empirical results of our algorithms on two domains: *citation* and *advertisements*. Both problems are modeled with a simple token-based HMM. We stress that token-based HMM cannot represent many of the constraints we used. The $d(y, \nu_{C(x)})$ function we used was an approximation of a Hamming distance function, which we discuss in section 8.5.3. All the constraints are given the same

[2]The logarithm is due to the fact that in our model, $\boldsymbol{\lambda} F(x,y)$ corresponded to a log-likelyhood of an HMM model, and ρ corresponded a crude estimation of log probability that a constraint does not hold

size	Inf.	sup.	**H**	**H+N**	**H+N+C** (Top-1)	**H+N+C** (Top-k)
5	no I	55.1	60.9	63.6	70.6	71.0
	I	66.6	69.0	72.5	76.0	77.8
10	no I	64.6	66.8	69.8	76.5	76.7
	I	78.1	78.1	81.0	83.4	83.8
15	no I	68.7	70.6	73.7	78.6	79.4
	I	81.3	81.9	84.1	85.5	86.2
20	no I	70.1	72.4	75.0	79.6	79.4
	I	81.1	82.4	84.0	86.1	86.1
25	no I	72.7	73.2	77.0	81.6	82.0
	I	84.3	84.2	86.2	87.4	87.6
300	no I	86.1	80.7	87.1	88.2	88.2
	I	92.5	89.6	93.4	93.6	93.5

Table 8.3: Experimental results for extracting fields from citations. Adding constraints improves the performance both during learning and inference.

penalty score ρ. The parameters ρ are tuned on the development set. We choose ρ to be $-\log 0.0001$ for citation and $-\log 0.0001$ for advertisements, while $\gamma = 0.1$ were fixed for all the experiments. The constraints for the two domains are listed in Table 8.1 and Table 8.2.

In both domains we ran 5 iterations for each of the semi-supervised learning algorithms. We compared the performance for training sets of size varying from 5 to 300 for the citations and from 5 to 100 for the advertisements we always tested on 100 held out examples. We ran 5 experiments in each setting, randomly choosing the training set. The results reported below are the averages over these 5 runs. In all semi-supervised experiments, 1000 unlabeled examples are used. Note that testing data is not included in the unlabeled data pool.

In our experiments, we verified the hypothesis that using constraints will improves the learned parameter, $\boldsymbol{\lambda}$. Moreover, we show that semi-supervised learning without guiding is beneficial when the labeled data is small, but can deteriorate the results when the training set is large. Furthermore, the method for weighting the effect of unlabeled data described in [101] is very useful, even when the training set is large. In other words, nonzero value of γ will have a positive effect in semi-supervised learning.

The experimental result for the two domains are given in Tables 8.4 and 8.3. To verify our claims we implemented several baselines, and compared their performance when the inference was done with (*I*) and without (*no I*) constraints. The first baseline, is the supervised model denoted by *sup*. The second baseline was a simple *top-1 Hard EM*[3] (denoted by **H** for Hard). The results in the fourth column of tables show that indeed, with very little training data, **H** improves on **sup**, but when the size of the training set increases, it deteriorates the results. The third baseline was *top-1 Hard EM*, but with weighting of the unlabeled model

[3]We also experimented with traditional EM without constraints, but the results were generally worse.

size	Inf.	sup.	**H**	**H+N**	**H+N+C** (Top-1)	**H+N+C** (Top-k)
5	no I	55.2	61.8	60.5	66.0	66.0
	I	59.4	65.2	63.6	69.3	69.6
10	no I	61.6	69.2	67.0	70.8	70.9
	I	66.6	73.2	71.6	74.7	74.7
15	no I	66.3	71.7	70.1	73.0	73.0
	I	70.4	75.6	74.5	76.6	76.9
20	no I	68.1	72.8	72.0	74.5	74.6
	I	71.9	76.7	75.7	77.9	78.1
25	no I	70.0	73.8	73.0	74.9	74.8
	I	73.7	77.7	76.6	78.4	78.5
100	no I	76.3	76.2	77.6	78.5	78.6
	I	80.4	80.5	81.2	81.8	81.7

Table 8.4: Experimental results for extracting fields from advertisements. Adding constraints improves the performance both during learning and inference.

(denoted as **H+N**). As expected, comparing columns 3 and 5 in the tables shows that this protocol always outperforms the supervised one, even when the training set is large. We then compared these baselines to our proposed protocol **H+N+C**, where we added the constraints to guide the **H+N** protocol and saw a significant improvement. We experimented with two flavors of the algorithm: the top-1 and the top-K version and did not observe large differences in performance.

An important comparison is to see what is the contribution of constraints in inference and in learning. We consistently observed that *I* consistently outperformed *no I* by large margin, but looking at the *I* rows shows that the learning protocol we propose does improve the performance compared to the **H+N** protocol and then adding the constraints on top of it. Figure 8.3 compares two protocols on the citation domain: **H+N+I** where we first run the **H+N** protocol and then apply the constraints during inference, and the **H+N+C+I** protocol that guides the **H+N** with constraints during learning. When there is a lot of labeled data, doing inference with constraints is more important than using constraints during learning. However, it is critical to use **H+N+C** when the amount labeled data is small. We also checked what is the combined effect of using constraints with semisupervised learning. Figure 8.4 shows that with just 20 labeled samples, our protocol achieves similar performance to the supervised version without constraints that was trained on 300 labeled samples.

In order to compare our work to [54], we run the experiments following this setting. [4]

[4]We found that the preprocessing procedure of [54] is slightly different ours.

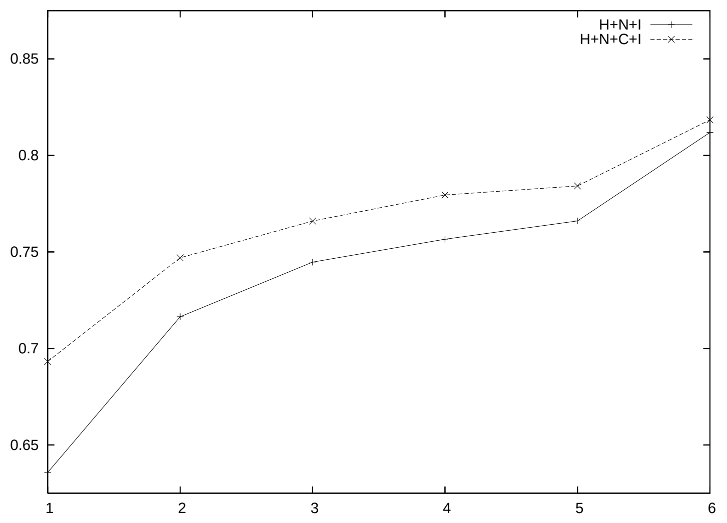


Figure 8.3: Comparison between **H+N** and **H+N+C** when both protocols use constraints for inference after learning. When there is a lot of labeled data, doing inference with constraints is more important than using constraints during learning. However, it is critical to use **H+N+C** when the amount labeled data is small.

8.7 Conclusions

We proposed to use constraints as a way to guide semi-supervised learning. The framework developed is general both in terms of the representation and expressiveness of the constraints, and in terms of the underlying model being learned – HMM in the current implementation. Moreover, our framework is a useful tool when the domain knowledge cannot be expressed by the model.

The results show that constraints can not only improve the performance of the final inference stage but also propagate useful information during the semi-supervised learning process and that training with the constraints is especially significant when the number of labeled training data is small.

Our future work proceeds in two directions. Given that the framework allows us to train the model discriminatively, we plan to do that, and extend to estimate the constraints' penalties too. On the application side, we would like to apply the framework to relation extraction where discriminative models has been proved to be very useful.

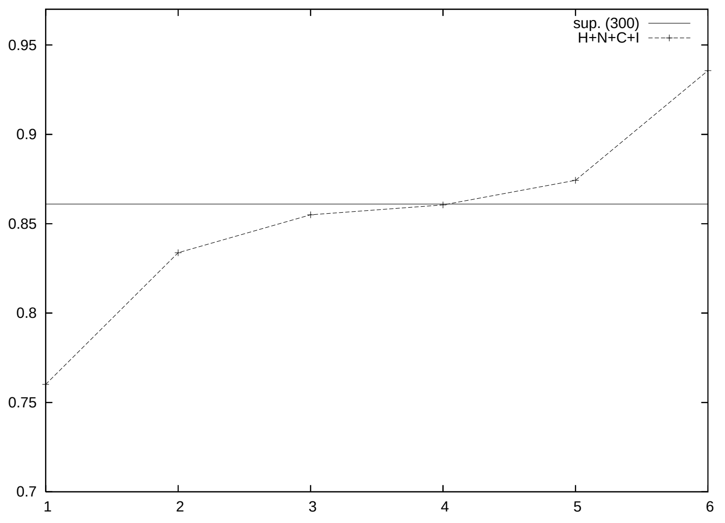


Figure 8.4: With 20 labeled examples our algorithm performs competatively to the supervised version trained on 300 smaples.

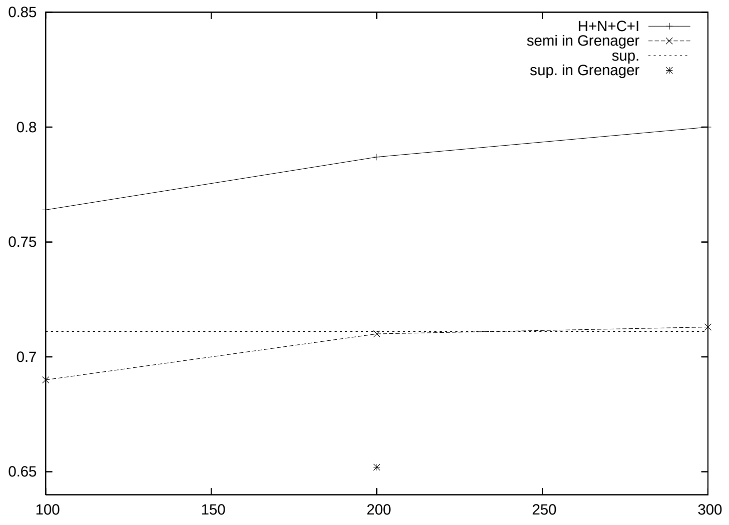


Figure 8.5: Comparison to results reported in Grenager et al. 2005.

Chapter 9

Conclusion and Discussion

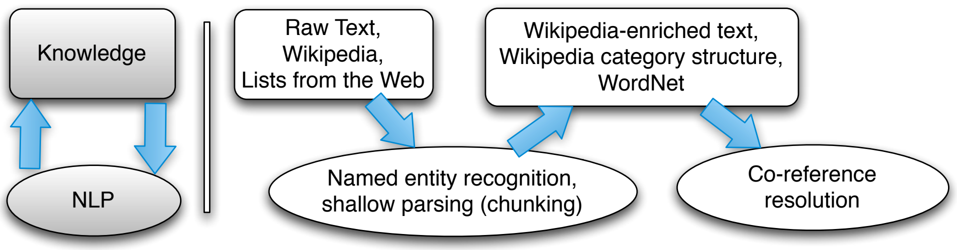

Knowledge
NLP
Raw Text, Wikipedia, Lists from the Web
Wikipedia-enriched text, Wikipedia category structure, WordNet
Named entity recognition, shallow parsing (chunking)
Co-reference resolution

Figure 9.1: The co-dependency between knowledge and NLP throughout this Thesis.

In this thesis we have argued that it is essential to use knowledge in NLP, proposed several ways of doing it, and provided case studies on several fundamental NLP problems. Understanding natural language often involves answering five types of "Wh" questions: who, what, where, when and why. We have focused on the first three questions (who, what, and where), which seem to be of a particular importance. We have tackled the problem of answering the "wh" questions in text as a series of four increasingly complex NLP tasks: text categorization, information extraction, concept disambiguation and co-reference resolution. We showed how to use knowledge encoded in form of constraints to guide semi-supervised learning for information extraction in Chapter 8. We showed how to use Wikipedia to bridge between information retrieval and text categorization in Chapter 3. Namely, in text categorization, we treat category names as queries and assign them a semantic interpretation using Wikipedia. This allows us to treat text categorization much like information extraction: we can categorize text to an unlimited number of categories without requiring training data.

However, the most significant contribution of this thesis is a successful symbiosis between injecting knowledge into text in order to improve NLP tools, and using better NLP tools in order to inject better knowledge into text. In [112] and in [145] we show how to use knowledge to improve an NER system. We use the resulting NER system to inject better knowledge into text by disambiguating important concepts to Wikipedia [111]. Finally, we take advantage of the injected knowledge to improve a co-reference NLP system in [113]. The summary of the mutual reinforcement loop between knowledge and NLP is summarized in Figure 9.1.

We have successfully used several types of knowledge throughout this thesis:

1. Data sparsity reduction methods based on analyzing large quantities of unlabeled text (Chapters 5 and 4).
2. Enriching text via encyclopedic knowledge. This can be done either by assigning topic-specific information (Chapter 3) or by cross-linking expressions in text to encyclopedic resources (Chapter 7).
3. Using first-order-logic-like constraints to guide self-training of a statistical model (Chapter 8).

There are several directions for future work:

1. It would be very interesting and extremely challenging to refine our understanding on the following questions. What would be a general way to come up with soft constraints for a given problem? What properties of a knowledge resource make it more amenable for use in NLP applications? When a new application is being developed, what is the general roadmap for designing knowledge-rich features within a statistical solution?
2. It is interesting to see whether it is possible to close the loop in Figure 9.1. For example, whether a better co-reference resolution system would allow better disambiguation to Wikipedia or whether a better system for disambiguation to Wikipedia would allow a better NER system.
3. It would be interesting to develop a general method for inducing task-specific word representations. For example, developing an approach which would be able to induce the Brown clusters while capturing gender and nationality in the co-reference setting, and sentiment (positive adjectives separate from negative adjectives) for sentiment analysis.

References

[1] R. K. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *ACL*, 2005.

[2] A. Bagga and B. Baldwin. Algorithms for Scoring Coreference Chains. In *MUC7*, 1998.

[3] C. Baker, C. Fillmore, and B. Cronin. The Structure of the Framenet Database. *International Journal of Lexicography*, 16(3):1–16, 2003.

[4] M. Banko, M. J. Cafarella, S. Soderl, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[5] Y. Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

[6] Y. Bengio, R. Ducharme, and P. Vincent. A Neural Probabilistic Language Model. In *NIPS*, 2001.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum Learning. In *ICML*, 2009.

[8] Y. Bengio, Duchar R., Pascal V., and Christian J. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003.

[9] Y. Bengio and J. Sénécal. Quick Training of Probabilistic Neural Nets by Importance Sampling. In *AISTATS*, 2003.

[10] E. Bengtson and D. Roth. Understanding the Value of Features for Coreference Resolution. In *EMNLP*, 2008.

[11] T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. Painless unsupervised learning with features. In *ACL*, 2010.

[12] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[13] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[14] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[15] R. C. Bunescu and M. Pasca. Using Encyclopedic Knowledge for Named entity Disambiguation. In *EACL*, 2006.

[16] M. Candito and B. Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, 2009.

[17] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, and T. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, 2010.

[18] X. Carreras, L. Màrquez, and L. Padró. Learning a Perceptron-Based Named Entity Chunker via Online Recognition Feedback. In *CoNLL*, 2003.

[19] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, 2006.

[20] M. Chang, L. Ratinov, and D. Roth. Guiding Semi-Supervision with Constraint-Driven Learning. In *ACL*, 2007.

[21] M. Chang, L. Ratinov, D. Roth, and V. Srikumar. Importance of Semantic Represenation: Dataless Classification. In *AAAI*, July 2008.

[22] Z. Chen, S. Tamang, A. Lee, X. Li, W. Lin, J. Artiles, M. Snover, M. Passantino, and H. Ji. CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. In *Text Analytics Conference*, 2010.

[23] H. Chieu and H. T. Ng. Named Entity Recognition with a Maximum Entropy Approach. In *CoNLL*, 2003.

[24] W. W. Cohen. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *KDD*, 2004.

[25] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

[26] M. Collins and T. Koo. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–70, 2005.

[27] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *SIGDAT*, 1999.

[28] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, 2008.

[29] S. Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, 2007.

[30] A. Culotta, M. Wick, R. Hall, and A. McCallum. First-Order Probabilistic Models for Coreference Resolution. In *NAACL*, 2007.

[31] H. Daumé. Frustratingly Easy Domain Adaptation. In *ACL*, 2007.

[32] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, June 2009.

[33] H. Daumé and D. Marcu. Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction. In *ICML*, 2005.

[34] H. Daumé and D. Marcu. Domain Adaptation for Statistical Classifiers. *Journal of Artficial Intelligence Research*, 26:101–126, 2006.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society, series B*, 39(1):1–38, 1977.

[36] K. Deschacht and M. Moens. Semi-supervised Semantic Role Labeling Using the Latent Words Language Model. In *EMNLP*, 2009.

[37] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. Robust, Light-weight Approaches to compute Lexical Similarity. Technical report, University of Illinois at Urbana-Champaign, 2009.

[38] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2. edition, 2001.

[39] S. Dumais, G. Furnas, T. Landauer, S. Deerwester, and R. Harshman. Using Latent Semantic Analysis To Improve Access To Textual Information. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 281–285. ACM, 1988.

[40] J. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:781–799, 1993.

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

[42] A. Fader, S. Soderland, and O. Etzioni. Scaling Wikipedia-based Named Entity Disambiguation to Arbitrary Web Text. In *WikiAI (IJCAI workshop)*, 2009.

[43] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[44] T. Finin, Z. Syed, J. Mayfield, P. McNamee, and C. Piatko. Using Wikitology for Cross-Document Entity Coreference Resolution. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009.

[45] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL*, 2005.

[46] J.R. Firth. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis*, pages 1–32, 1957.

[47] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named Entity Recognition through Classifier Combination. In *CoNLL*, 2003.

[48] Y. Freund and R. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296, 1999.

[49] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *IJCAI*, 2005.

[50] E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*, 2007.

[51] E. Gabrilovich and S. Markovitch. Harnessing the Expertise of 70,000 Human Editors: Knowledge-Based Feature Generation for Text Categorization. *Journal of Machine Learning Research*, 8:2297–2345, 2007.

[52] Y. Goldberg and M. Elhadad. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *NAACL*, 2010.

[53] Y Goldberg, R. Tsarfaty, M. Adler, and M. Elhadad. Enhancing Unlexicalized Parsing Performance Using a Wide Coverage Lexicon, Fuzzy Tag-Set Mapping, and EM-HMM-Based Lexical Probabilities. In *EACL*, 2009.

[54] T. Grenager, D. Klein, and C. Manning. Unsupervised Learning of Field Segmentation Models for Information Extraction. In *ACL*, 2005.

[55] A. Haghighi and D. Klein. Prototype-Driven Grammar Induction. In *ACL*, 2006.

[56] A. Haghighi and D. Klein. Simple Coreference Resolution with Rich Syntactic and Semantic Features. In *EMNLP*, 2009.

[57] A. Haghighi and D. Klein. Coreference Resolution in a Modular, Entity-Centered Model. In *ACL*, 2010.

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

[59] Z. Harris. Distributional structure. In J. J. Katz, editor, *Philosophy of Linguistics*. New York: Oxford University Press, 1985.

[60] T. Honkela. Self-organizing maps of words for natural language processing applications. In *Proceedings of the International ICSC Symposium on Soft Computing*, 1997.

[61] T. Honkela, V. Pulkki, and T. Kohonen. Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. In *ICANN*, 1995.

[62] F. Huang and A. Yates. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL*, 2009.

[63] L. Huang. Forest reranking: Discriminative parsing with non-local features. In *ACL*, 2008.

[64] F. Jelenek and R. Mercer. Interpolated Estimation of Markov Source Parameters from Sparse Data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[65] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *ACL*, 2007.

[66] T. Joachims, T. Hofmann, Y. Yue, and C. Yu. Predicting Structured Objects with Support Vector Machines. *Communications of the ACM, Research Highlight*, 52(11):97–104, November 2009.

[67] S. Kaski. Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering. In *IJCNN*, 1998.

[68] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987.

[69] J. Kazama and K. Torisawa. A New Perceptron Algorithm for Sequence Labeling with Non-Local Features. In *EMNLP-CoNLL*, 2007.

[70] J. Kazama and K. Torisawa. Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *EMNLP*, 2007.

[71] R. Khardon, D. Roth, and Servedio R. Efficiency versus Convergence of Boolean Kernels for On-Line Learning Algorithms. In *NIPS*, 2001.

[72] T. Koo, X. Carreras, and M. Collins. Simple Semi-supervised Dependency Parsing. In *ACL*, 2008.

[73] V. Krishnan and C. D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL*, 2006.

[74] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in web text. In *KDD'09*, 2009.

[75] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 2001.

[76] T Landauer, P. Foltz, and D. Laham. An Introduction to Latent Semantic Analysis. *Discourse Processes*, (25):259–284, 1998.

[77] K. Lang. NewsWeeder: Learning to Filter Netnews. In *ICML*, 1995.

[78] D. B. Lenat. CYC: a large-scale investment in knowledge infrastructure. *Communications of ACM*, 38:33–38, November 1995.

[79] Wei Li and Andrew McCallum. Semi-supervised sequence modeling with syntactic topic models. In *AAAI*, 2005.

[80] P. Liang. Semi-Supervised Learning for Natural Language. *Masters thesis, Massachusetts Institute of Technology*, 2005.

[81] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *ACL-IJCNLP*, 2009.

[82] C. Lund, K.and Burgess and R. Atchley. Semantic and associative priming in high-dimensional semantic space. In *Cognitive Science Proceedings, LEA*, 1995.

[83] K Lund and C. Burgess. Producing highdimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208, 1996.

[84] X. Luo. On coreference resolution performance metrics. In *HLT*, 2005.

[85] M. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*, 19:313–330, June 1993.

[86] S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37, 1998.

[87] J. Mayfield and et al. Cross-Document Coreference Resolution: A Key Technology for Learning by Reading. In *Proceedings of the AAAI 2009 Spring Symposium on Learning by Reading and Learning to Read*, March 2009.

[88] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *ICML*, 1998.

[89] B. Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20:155–171, June 1994.

[90] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, 2007.

[91] S. Miller, J. Guinness, and A. Zamanian. Name Tagging with Word Clusters and Discriminative Training. In *NAACL*, 2004.

[92] D. Milne and I. H. Witten. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *In Proceedings of AAAI 2008*, 2008.

[93] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM*, 2008.

[94] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *ICML*, 2007.

[95] A. Mnih and G. Hinton. A Scalable Hierarchical Distributed Language Model. In *NIPS*, 2009.

[96] F. Morin and Y. Bengio. Hierarchical Probabilistic Neural Network Language Model. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS*, 2005.

[97] V. Nastase and M. Strube. Decoding wikipedia categories for knowledge acquisition. In *AAAI*, 2008.

[98] A. Ng and M. Jordan. On Discriminative vs. Generative classifiers: A comparison of logistic regression and Naive Bayes. In *NIPS*, 2001.

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

[100] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *ICML*, 2005.

[101] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39:103–134, May 2000.

[102] NIST. The ACE Evaluation Plan. www.nist.gov/speech/tests/ace/index.htm, 2004.

[103] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *ACL*, 1993.

[104] S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *ACL*, 2006.

[105] V. Punyakanok and D. Roth. The Use of Classifiers in Sequential Inference. In *NIPS*, 2001.

[106] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and Inference over Constrained Output. In *IJCAI*, 2005.

[107] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE*, 1989.

[108] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. A multi-pass sieve for coreference resolution. In *EMNLP*, 2010.

[109] A. Rahman and V. Ng. Coreference Resolution withWorld Knowledge. In *ACL*, 2011.

[110] R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *ICML*, 2006.

[111] L. Ratinov, D. Downey, M. Anderson, and D. Roth. Local and Global Algorithms for Disambiguation to Wikipedia. In *ACL*, 2011.

[112] L. Ratinov and D. Roth. Design Challenges and Misconceptions in Named Entity Recognition. In *CoNLL*, 2009.

[113] L. Ratinov and D. Roth. Learning-based Multi-Sieve Co-reference Resolution with Knowledge. In *review at EMNLP*, 2012.

[114] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*, 1999.

[115] A. Ritter, Mausam, and O. Etzioni. A latent dirichlet allocation method for selectional preferences. In *ACL*, 2010.

[116] H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biological Cybernetics*, (61):241–254, 1989.

[117] N. Rizzolo and D. Roth. Modeling Discriminative Global Inference. In *ICSC*, 2007.

[118] D. Roth. Learning to Resolve Natural Language Ambiguities: A Unified Approach. In *AAAI*, 1998.

[119] D. Roth. Learning in Natural Language. In *IJCAI*, 1999.

[120] D. Roth and W. Yih. Integer Linear Programming Inference for Conditional Random Fields. In *ICML*, 2005.

[121] D. Roth and D. Zelenko. Part of Speech Tagging Using a Network of Linear Separators. In *COLING-ACL*, 1998.

[122] Noah A. S. and Jason E. Contrastive Estimation: Training Log-Linear Models on Unlabeled Data. In *ACL*, 2005.

[123] M. Sahlgren. Vector-Based Semantic Analysis: Representing Word Meanings Based on Random Labels. In *Proceedings of the Semantic Knowledge Acquisition and Categorisation Workshop, ESSLLI*, 2001.

[124] M. Sahlgren. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*, 2005.

[125] M. Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm University, 2006.

[126] E. T. Sang and S. Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *CoNLL00*, 2000.

[127] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis. Learning first-order Horn clauses from web text. In *EMNLP*, 2010.

[128] F. Sha and F. Pereira. Shallow Parsing with Conditional Random Fields. In *NAACL*, 2003.

[129] H. Shen and A. Sarkar. Voting Between Multiple Data Representations for Text Chunking. *Advances in Artificial Intelligence*, 2005.

[130] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *AAAI*, 2005.

[131] P. Singh, T. Lin, E.T. Mueller, G. Lim, T. Perkins, and W.L. Zhu. Open Mind Common Sense: Knowledge Acquisition from the General Public. In *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, 2002.

[132] V. Spitkovsky, h. Alshawi, and D. Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *NAACL*, 2010.

[133] M. Strube and S. P. Ponzetto. WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *AAAI*, 2006.

[134] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

[135] J. Suzuki and H. Isozaki. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *ACL*, 2008.

[136] J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. An empirical study of semi-supervised structured conditional models for dependency parsing. In *EMNLP*, 2009.

[137] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[138] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *SIGDAT*, 2002.

[139] J. Thorsten. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *ICML*, pages 143–151, 1997.

[140] K. E. Tjong and F. De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL*, 2003.

[141] A. Toral and R. Munoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. In *EACL*, 2006.

[142] K. Toutanova, A.Haghighi, and C. Manning. Joint Learning Improves Semantic Role Labeling. In *ACL*, 2005.

[143] K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, 2003.

[144] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[145] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, 2010.

[146] J. Turian, L. Ratinov, Y. Bengio, and D. Roth. A preliminary evaluation of word representations for named-entity recognition. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009.

[147] P. Turney and P. Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 2010.

[148] A. Ushioda. Hierarchical clustering of words. In *COLING*, 1996.

[149] W. Yih V. Punyakanok, D. Roth and D. Zimak. Learning and Inference over Constrained Output. In *IJCAI*, 2005.

[150] D. Vadas and J. R. Curran. Parsing noun phrase structure with CCG. In *ACL*, 2008.

[151] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[152] V. Vapnik and A. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[153] J. Väyrynen and T. Honkela. Word Category Maps based on Emergent Features Created by ICA. In *Proceedings of the STeP'2004 Cognition + Cybernetics Symposium*, 2004.

[154] J. Väyrynen and T Honkela. Comparison of Independent Component Analysis and Singular Value Decomposition in Word Context Analysis. In *AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, 2005.

[155] J. Väyrynen, T. Honkela, and L. Lindqvist. Towards explicit semantic features using independent component analysis. In *Proceedings of the Workshop Semantic Content Acquisition and Representation (SCAR)*, 2007.

[156] J. Veenstra. Representing text chunks. In *EACL*, 1999.

[157] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A Model-Theoretic Coreference Scoring Scheme. In *MUC6*, 1995.

[158] R. Vilalta and I. Rish. A decomposition of classes via clustering to explain and improve naive bayes. In *ECML*, 2003.

[159] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC*, 2010.

[160] M. Wick, K. Rohanimanesh, K. Bellare, A. Culotta, and A. McCallum. SampleRank: Training Factor Graphs with Atomic Gradients. In *ICML*, 2011.

[161] F. Wu, R. Hoffmann, and D. Weld. Information extraction from Wikipedia: moving down the long tail. In *KDD*, 2008.

[162] H. Xianpei and Z. Jun. Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. In *CIKM*, 2009.

[163] T. Zhang and D. Johnson. A Robust Risk Minimization based Named Entity Recognition System. In *CoNLL*, 2003.

[164] H. Zhao, W. Chen, C. Kit, and G. Zhou. Multilingual dependency learning: a huge feature engineering method to semantic dependency parsing. In *CoNLL*, 2009.

[165] Y. Zhou, L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney. Resolving Surface Forms to Wikipedia Topics. In *COLING*, 2010.

[166] X. Zhu. Semi-Supervised Learning Literature Survey, 2006.