EXPLOITING DOMAIN AND TASK REGULARITIES FOR ROBUST NAMED ENTITY RECOGNITION

Andrew O. Arnold

August 2009
CMU-ML-09-109

School of Computer Science
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
William W. Cohen, *Chair*
Tom M. Mitchell
Noah A. Smith
ChengXiang Zhai (University of Illinois at Urbana-Champaign)

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

This research was sponsored by the National Institute of Health under contract no. 1R01GM081293, the National Institute of Health under contract no. 1R01GM078622-01, SRI International under contract no. 71000152, SRI International under contract no. 0300021110, SRI International under contract no. 03-000211/TASK7, and the National Science Foundation under contract no. REC-0326153. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: machine learning, named entity extraction, transfer learning

For my parents

Abstract

It is often convenient to make certain assumptions during the learning process. Unfortunately, algorithms built on these assumptions can often break down if the assumptions are not stable between train and test data. Relatedly, we can do better at various tasks (like named entity recognition) by exploiting the richer relationships found in real-world complex systems. By exploiting these kinds of non-conventional regularities we can more easily address problems previously unapproachable, like transfer learning. In the transfer learning setting, the distribution of data is allowed to vary between the training and test domains, that is, the independent and identically distributed (i.i.d.) assumption linking train and test examples is severed. Without this link between the train and test data, traditional learning is difficult.

In this thesis we explore learning techniques that can still succeed even in situations where i.i.d. and other common assumptions are allowed to fail. Specifically, we seek out and exploit regularities in the problems we encounter and document which specific assumptions we can drop and under what circumstances and still be able to complete our learning task. We further investigate different methods for dropping, or relaxing, some of these restrictive assumptions so that we may bring more resources (from unlabeled auxiliary data, to known dependencies and other regularities) to bear on the problem, thus producing both better answers to existing problems, and even being able to begin addressing problems previously unanswerable, such as those in the transfer learning setting.

In particular, we introduce four techniques for producing robust named entity recognizers, and demonstrate their performance on the problem domain of protein name extraction in biological publications:

- **Feature hierarchies** relate distinct, though related, features to one another via a natural linguistically-inspired hierarchy.
- **Structural frequency features** exploit a regularity based on the structure of the data itself and the distribution of instances across that structure.
- **Snippets** link data not by the distribution of the instances or their features, but by their labels. Thus data that have different attributes, but similar labels, will be joined together, while instances that have similar features, but distinct labels, are segregated to allow for variation between domains.
- **Graph relations** represent the entities contained in the data and their relationships to each other as a network which is exploited to help discover robust regularities across domains.

Thus we show that learned classifiers and extractors can be made more robust to shifts between the train and test data by using data (both labeled and unlabeled) from related domains and tasks, and by exploiting stable regularities and complex relationships between different aspects of that data.

Acknowledgments

I would like to thank my advisor, William W. Cohen, for all his support and guidance throughout the course of our research together, always treating me as a colleague and helping me keep my eye on the big picture. He took a chance on me for which I will always be grateful.

I would also like to thank The National Institutes of Health's grant R01 GM078622, without whose support much of this work would have been impossible.

To my committee: thank you for supporting me through the entire thesis process. Your comments and suggestions have demonstrably improved not only the work, but also the way I view my research and its (possible) contribution to the greater mission of science.

To the Machine Learning Department: thank you for giving me the opportunity to further myself, and be part of this exciting new field. Diane Stidle always provided a smile and free food, along with invaluable information and perspective.

To all my collaborators, mentors and comrades in research: thank you for your insights, suggestions, inspirations and good company. Richard Scheines and Joseph E. Beck helped me discover who I was as a researcher, and Naoki Abe made me feel like the ideas I had kicking around in my head might be of some value to others around me. Robert F. Murphy, Saboor A. Sheikh and the entire SLIF team gave me an amazing application in which I could test my ideas and priceless feedback, while Zhenzhen Kou, Einat Minkov, Richard Wang and the rest of the Text Learning Group gave me a forum in which to share ideas. Hang Li and all the interns at Microsoft Reasearch Asia showed me you could do research and have fun at the same time, while Ramesh Nallapati always had a good idea and kind word.

To all my teachers: you always suffered my questions and challenged me to do my best. Simon Lok, thank you for introducing me to computer science and presenting it in a way that excited both my intellect and creativity. John R. Kender, thank you for showing me the fulfillment of teaching. And Eleazar Eskin, thank you for taking me under your wing and helping me in my first research dabblings.

To all the selfless people at the intitutions of learning that have taken me in (Mirman, Harvard-Westlake, Columbia, Carnegie Mellon), thank you so much! Without your sacrifice and support, I could never be where I am today. Especially Mirman, which, along with my parents, taught me that it was OK to be a bit different.

To all the unsung heroes of television:, thank you for giving a nerdy kid like me a place to learn and have fun after school: James Burke, Bill Nye, Paul Zaloom and Don Herbert. Thank you PBS for *Newton's Apple*, *Square One*, *3-2-1 Contact* and *Nova*. And, of course, to the sung heroes as well: Aristotle,

Francis Bacon, Galileo, Isaac Newton, Charles Darwin and Richard Feynman. The hope of one day being as cool as any one of you has kept me going.

To my friends and roommates: thank you for making the past five years bearable, and the next fifty surely filled with friendship and comraderie. Thomas LaToza, Jure Leskovec, and Brian Ziebart put up with funky foods and an occult sleeping schedule, while Hao Cen proved an invaluable sounding board and formidable raquetball opponent.

Grandma and Aunt Lucia: you have provided me the push to strive for my goals, and the love to know that everything will be OK even if I fall short.

Josh and Diana: you have fulfilled your roles as siblings perfectly, and I am so proud of how our relationships have grown and developed as we move from kids to grown-ups together.

Mom and Dad, you knew the time. Besides a great set of genes, you instilled in me a confidence to be myself, even when I didn't fit in. This, along with your patience with my incessant 'whys' has kept the flames of a child's natural curiosity stoked and made everything in life so much more interesting.

Finally, I would like to thank Michelle: while it is *conceivable* that I might have somehow managed to navigate my way through the past four years without you, even if I had, I would still be utterly lost.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The desire to exploit information attained from previous effort, and not to start each new endeavor *de novo* is perhaps part of human nature, and certainly a maxim of the scientific method. Nevertheless, due to the difficulty of integrating knowledge from distinct, but related, experimental *domains* (the distribution from which the data is drawn) and *tasks* (the type of prediction desired from the learner), it is common practice in most machine learning studies to focus on training and tuning a model to a single, particular domain and task pair, or *setting*, at the expense of all others. Often, once work has completed on one setting, the researcher begins afresh on the next, carrying over only the techniques and experience learned, but often not the data or model itself.

Consider the task of *named entity recognition* (NER). Specifically, suppose you are given a corpus of encyclopedia articles in which all the personal name mentions have been labeled. The standard supervised machine learning problem is to learn a classifier over this training data that will successfully label unseen test data drawn from the same distribution as the

training data, where "same distribution" could mean anything from having the train and test articles written by the same author to having them written in the same language. Having successfully trained a named entity classifier on this encyclopedia data, now consider the problem of learning to classify tokens as names in instant messenger data. Clearly the problems of identifying names in encyclopedia articles and instant messages are closely related, and learning to do well on one should help your performance on the other. At the same time, however, there are serious differences between the two problems that need to be addressed. For instance, capitalization, which will certainly be a useful feature in the encyclopedia problem, may prove less informative in the instant messenger data since the rules of capitalization are followed less strictly in that domain. Thus there seems to be some need for altering the classifier learned on the first problem (called the *source domain*) to fit the specifics of the second problem (called the *target domain*). This is the problem of *domain adaptation* [Daumé III and Marcu, 2006] and constitutes a subproblem in the broader field of **transfer learning,** which has been studied as such for at least the past ten years [Thrun, 1996; Baxter, 1997].

The intuitive solution seems to be to simply train on the target domain data. Since this training data would be drawn from the same distribution as the data you will ultimately test over, this approach avoids the transfer issue entirely. The problem with this idea is that often large amounts of labeled data are not available in the target domain. While it has been shown that even small amounts of labeled target data can greatly improve transfer results [Chelba and Acero, 2004; Daumé III, 2007], there has been relatively little work on the case when there is no labeled target data available, that is, totally unsupervised domain adaptation. In this scenario, one way to adapt a model trained on the source domain is to make the unlabeled *target test data* available to the model during training time. Leveraging unlabeled test data during training time is called *transductive learning* and is a well studied problem in the scenario when the training data and test data come from the same domain.

However, transduction is not well-studied in a transfer setting, where the training and test data come from different domains, which will be the learning scenario upon which we focus throughout most of the thesis.

Figures 2.1 and 1.1 give schematic overviews of the ways we see these techniques intersecting and overlapping with one another, while Table 2.1 provides a detailed breakdown of various transfer learning settings.

1.2 Goal of the thesis: robust learning

This thesis is concerned with various forms of robust learning both within and without the framework of transfer learning:

> **Regularities and relationships among various aspects of data can be exploited to help create classifiers that are more robust across the data as a whole (both source and target).**

1.2.1 Robust learning in the face of unstable properties

It is often convenient to make certain assumptions during the learning process. Unfortunately, algorithms built on these assumptions can often break down if the assumptions are not *stable* between train and test data. We define a property of the data to be *stable* if said property remains *relatively unchanged* across *variations* in other aspects of the data, where such properties can be attributes of the data instances themselves or relationships among different parts of the data; and the 'variations' allowed among the data and the degree to which the stable property must remain 'unchanged' is defined with respect to the degree of robustness desired. For instance, in traditional learning, given $(x, y)_{train}$ drawn from some

training distribution D_{train}, and $(x, y)_{test}$ drawn from some test distribution D_{test}, we assume that $p_{train}(y \mid x) = p_{test}(y \mid x)$. If we allow $p(y \mid x)$ to vary across training and testing data (that is, if we allow $D_{train} \neq D_{test}$, as in the domain adaptation setting), a standard machine learning technique like naive Bayes may fail. In the language of this thesis, this learning technique is not *robust* to this change in the data. *Our thesis is that we can make learned classifiers and extractors more robust by using data (both labeled and unlabeled) from related domains and tasks, and by exploiting stable regularities and complex relationships between different aspects of that data.*

1.2.2 Exploiting rich relationships

Relatedly, we can do better at various tasks (like information extraction) by exploiting the richer relationships found in real-world complex systems. When we start working with such a system, we usually find it convenient to first abstract away to a relatively simply stated learning problem, such as: *Given an example x, predict its label y.* This type of simplifying reduction is often necessary (at the expense of richer representations incorporating more domain knowledge and auxiliary sources of information) in order to frame the learning problem in a way that is consistent with the often harsh assumptions underlying many favored learning techniques. While these assumptions may be useful in providing structure in relatively simple learning problems, when faced with complex, real-world systems, they can often prove burdensome, or fail all together, and may actually be better replaced with problem-specific structure such as regularities among features or external sources of data.

1.2.3 Transfer learning

By exploiting these kinds of non-conventional regularities we can more easily address problems previously unapproachable, like transfer learning. In the transfer learning setting, the distribution of data is allowed to vary between the training and test domains, that is, the i.i.d. assumption linking train and test examples is severed. Without this link between the train and test data, traditional learning is difficult. Take, for example, the problem of training an extractor to identify the sender and recipient of a letter. For our training data we are given formal business letters with their senders and recipients labeled. For testing, however, we are required to identify the sender and recipient not in business letters but in student e-mails. Whereas in the non-transfer, business to business, learning case we could exploit regularities in the tokens themselves, for instance, looking for capitalized words that do not begin a sentence, in the transfer setting, this capitalization property may no longer hold between the train and test domains, that is, it is not stable. In light of this, we need a new relationship linking the domains together, an information path linking the training data to the test data. One possibility in this example would be to exploit the common structure of the letters themselves: specifically, the property of recipient names being located at the start of a letter, and sender names being located at the end. This tends to be true both in formal business letters and informal e-mails, and thus provides a stable regularity from which our classifier can generalize from the training data to the test data. In this way we can make use of one type of regularity (document structure) when another (the conditional distribution of capitalized names) ceases to hold.

Thus, in this thesis we try to find learning techniques that can still succeed even in situations where i.i.d. and other common assumptions are allowed to fail. Specifically, we seek out and exploit regularities in the problems we encounter and document which specific assumptions we can drop and under what circumstances and still be able to complete our learning

task. We further investigate different methods for dropping, or relaxing, some of these restrictive assumptions so that we may bring more resources (from auxiliary data, to known dependencies and other regularities) to bear on the problem, thus producing both better answers to existing problems, and even being able to begin addressing problems previously unanswerable, such as those in the transfer learning setting.

1.3 Scope of the thesis: named entity recognition (NER)

For most of this thesis we will focus on the specific problem of learning to extract protein names from articles published in biological journals. In the *named entity resolution* (NER) formalism, a document is segmented into a sequence of tokens, with each of these tokens[1] then being classified as belonging to one of a set of possible label classes – in our case, the binary set {PROTEIN, NON_PROTEIN}. A standard technique for this kind of problem is to gather a corpus of documents drawn from the domain on which you will eventually be evaluated. These documents then need to be painstakingly hand-labeled by a domain expert in order to identify which tokens in the document represent proteins, and which do not. The 'expertise' of this domain specialist should not be underestimated, since such biological distinctions are subtle and often elude all but the most experienced annotators. The work is therefore slow, and the resulting annotated datasets are often relatively small and expensive.

We have access to such a corpus of protein-labeled abstracts from biological articles. Several techniques have been proposed for building protein-name extractors over these abstracts and their performances have been evaluated with respect to extracting new proteins from other, previously unseen abstracts drawn from a similar distribution of articles [Franzén et al., 2002]. In our work, however, we are interested in identifying proteins, not in abstracts, but

[1]Multi-token entities, or **spans**, are possible, and in fact common, but we focus here on the single token entity example for ease of explanation.

in the captions of papers (we use this information to create a structured search engine of images and captions from biological articles [Murphy et al., 2004]). To this end we have downloaded tens of thousands of open-access, full text articles from the Internet. Unfortunately, all of these documents are wholly unlabeled and we do not have the resources to label them ourselves. Thus, our problem is: given labeled abstracts (source training domain) and unlabeled captions and full text (source auxiliary training data), how can we train a model that will extract proteins well from unseen captions (target test domain). This is at once a semi-supervised learning problem (due to the unlabeled auxiliary training data) [Zhu, 2005], and a domain adaptation problem (due to the difference in domains from which the source and target data are drawn).

1.4 Approach & organization of the thesis

Our thesis attempts to explore the ways we can relax assumptions and exploit regularities in order to better solve real-world learning problems. The following chapters introduce examples of problems involving violated assumptions, and the solutions we came up with for overcoming these broken assumptions. Figure 1.1 shows one way of visualizing the various types of structure and regularity that can be tapped in solving various learning problems. In this model, instances x, their labels y, and constituent features F, can be joined in various relationships. For instance, the standard assumption joining instances is that they are all drawn independently from an identical distribution (i.i.d.). In the problem we face, however, this assumption is violated as instances (words) are drawn from different sections of a document (abstract, caption, etc.) and therefore have different distributions within those sections. Therefore, in this setting the i.i.d. assumption linking the instances to each other (most importantly, linking the training instances to the test instances) is severed, resulting in training and testing sets of seemingly unrelated instances among which it appears impossible

to generalize. If we exploit a different regularity, however, re-linking the instances to each other in some way and taking the place of the invalidated i.i.d. assumption (see the top-left cloud in Figure 1.1), we are again able to learn and generalize across samples of training and test data.

In this thesis we explore four main approaches to solving this problem of robust named entity recognition:

1. When the assumption that instances share the same set of features fails to hold, we develop a new method for relating these distinct, though related, features to one another via a natural linguistically-inspired hierarchy (the bottom cloud in Figure 1.1). These are the **feature hierarchies** explained in Chapter 3.

2. Chapter 4 introduces what we call **structural frequency features**, a regularity based on the structure of the data itself and the distribution of instances across that structure. These are represented by the upper-left cloud in the diagram, linking instances of the data by their inherent structure.

3. Chapter 5 introduces **snippets**, represented by the upper-right cloud in the diagram, linking the data not by the distribution of the instances or their features, but rather by their labels. Thus data that have very different attributes, but similar labels, will be joined together, while instances that appear to have similar features, but distinct labels, are segregated to allow for variation between domains.

4. Finally, the top middle cloud in Figure 1.1 represents the graph relations of Chapter 6 wherein the different entities contained in the data and their relationships to each other are represented as a network which is exploited to help discover robust regularities across domains.

Chapter 2 goes into more detail concerning the various techniques that currently exist for robust learning, as summarized in Table 2.1 and Figure 2.1. A large amount of time is spent discussing *transfer learning* and its close relationship to the more general goal of this thesis, robustness. In particular, we relate transfer learning's goal of training learners that can generalize across data drawn from different distribution to our goal of producing robust classifiers that perform well across a variety of related data sources. Following that, we further explore the approaches introduced in this section (visually summarized in Figure 1.1) and show how they contribute to this thesis' goal of robust learning in real-world systems.

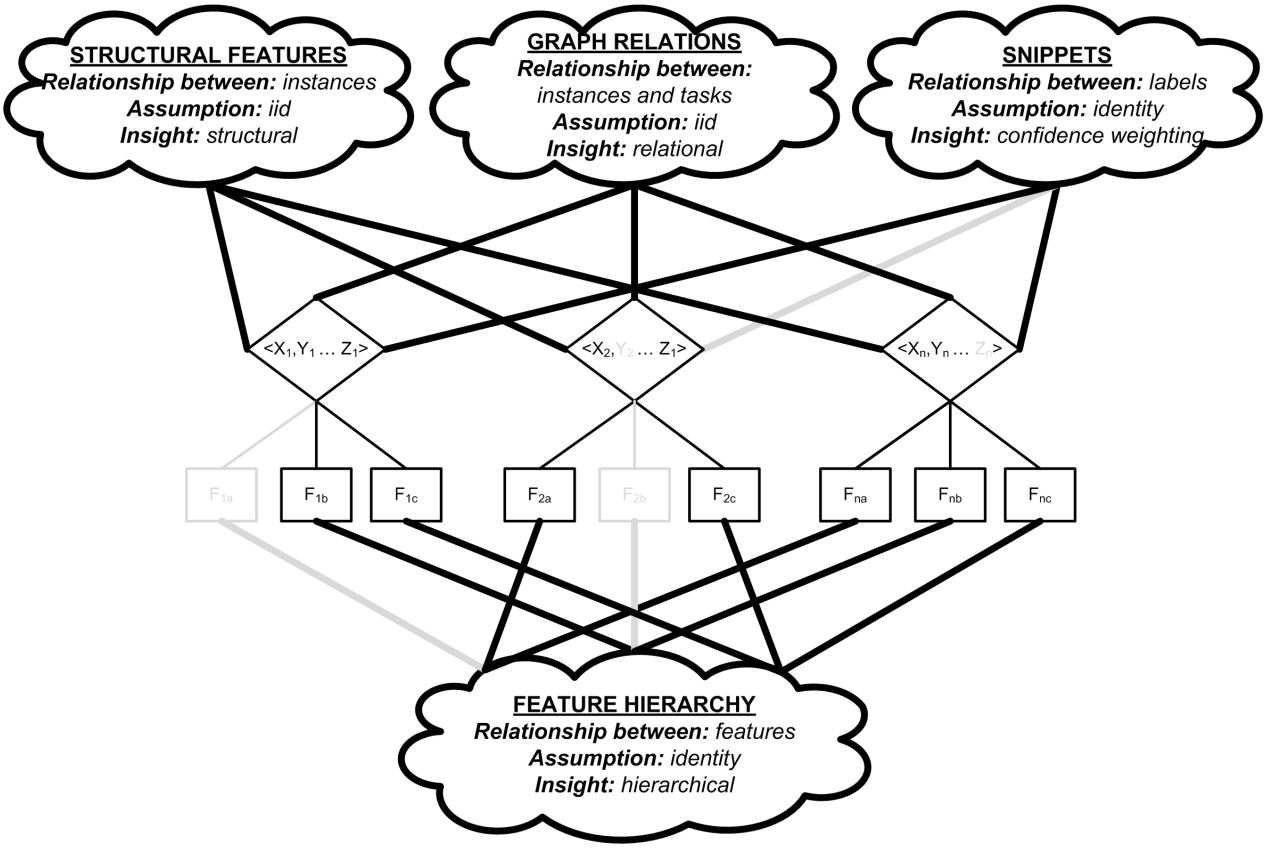


Figure 1.1: Visualization of the various types of structure used for robust learning. X's represent instances, while $Y...Z$'s represent different task labels for that instance. Dark lines denote observed variables and relationships, while light lines symbolize unobserved data. Paths between and among instances, features and labels are conducted via *clouds* representing common relationships between these attributes. These paths allow information to flow from one type of observation in a certain domain or task to other related, though possibly distinct, types of observations in related domains and tasks. For example, knowledge about one instance-label tuple $\langle x_1, y_1 \rangle$ can directly inform an observer about another, unseen label, y_2, due to the i.i.d. relationship between x_1 and x_2 and the stability of $p(y|x)$. Similarly, knowledge of x_1's value for feature b (F_{1b}) can help you estimate the value for the unobserved F_{1a} if there is some relationship (as in our hierarchical lexical features example) linking the features to each other. Relatedly, knowledge that instances x_1 and x_2 share a common label (z_1) for task Z, along with knowledge of x_2's Y label (y_2), might in turn help predict x_1's Y label (y_1). (For example, if x_1 and x_2 are instances of *abstracts*, Y's are their labeled *gene mentions*, and z_1 is an *author* they share in common.) In much of the work of this thesis these relationships are manifested as *external facts* and assumptions, for example, external linguistic knowledge about the hierarchy relating lexical features to one another, external biological knowledge constraining which proteins can occur in which regions of a cell, or external citation and authorship information as in the previous example. These external data sources can often provide the information paths necessary to link various aspects of the data together, allowing us to learn in complex settings where common assumptions, like i.i.d., may not hold.

Chapter 2

Survey

2.1 Current state of the art

Throughout this section you may refer to Figure 2.1 to get an overall view of the state of the art.

2.1.1 Transfer learning

The phrase **transfer learning** covers several different subproblems. When only the type of data being examined is allowed to vary (from news articles to e-mails, for example), the transfer problem is called *domain adaptation* [Daumé III and Marcu, 2006]. When the task being learned varies (say, from identifying person names to identifying protein names), the transfer problem is called *multi-task learning* [Caruana, 1997]. Both of these are considered specific types of the over-arching *transfer learning* problem, and both seem to require a way of altering the classifier learned on the first problem (called the *source domain*, or *source task*) to fit the specifics of the second problem (called the *target domain*, or *target task*).

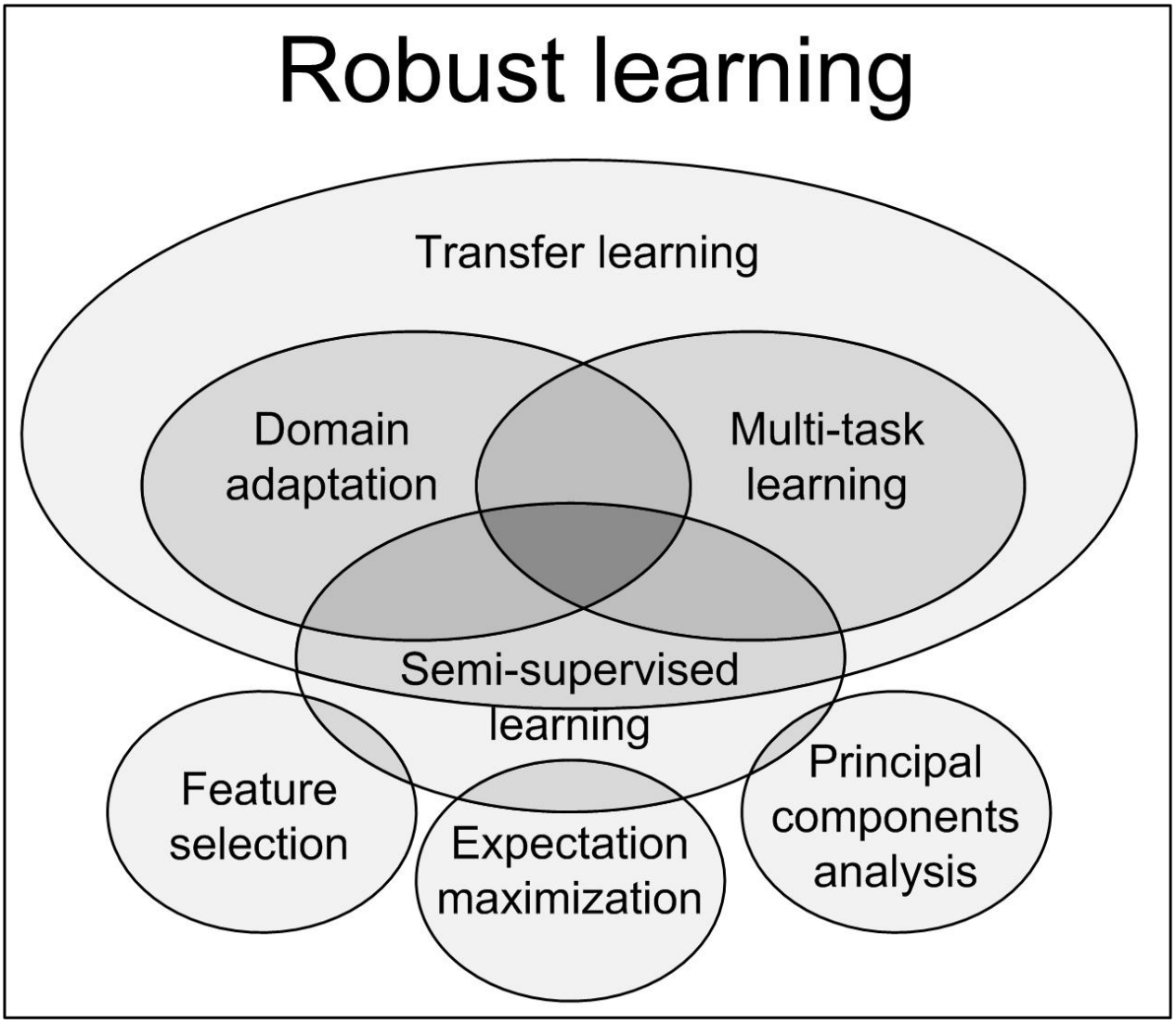


Figure 2.1: Venn diagram representation of the subspace of robust learning settings. Domain adaptation and multi-task learning are represented as subsets of transfer learning, which is itself a subset of all robust learning techniques. These techniques can also intersect with semi-supervised methods. A sampling of non-transfer robust learning techniques (such as sparse feature selection, expectation maximization and principal components analysis) are also included for completeness. Compare with Table 2.1, which structures the transfer learning sub-region into greater detail.

More formally, given an example x and a class label y, the standard statistical classification task is to assign a probability, $p(y|x)$, to x of belonging to class y. In the binary classification case the labels are $Y \in \{0, 1\}$. In the case we examine, each example x_i is represented as a vector of binary features $(f_1(x_i), \cdots, f_F(x_i))$ where F is the number of features. The data consists of two disjoint subsets: the training set $(X_{train}, Y_{train}) = \{(x_1, y_1) \cdots, (x_N, y_N)\}$, available to the model for its training and the test set $X_{test} = (x_1, \cdots, x_M)$, upon which we want to use our trained classifier to make predictions.

In the paradigm of *inductive learning*, (X_{train}, Y_{train}) are known, while both X_{test} and Y_{test} are completely hidden during training time. In this cases X_{test} and X_{train} are both assumed to have been drawn from the same distribution, $\mathcal{D}$. In the setting of *transfer learning*, however, we would like to apply our trained classifier to examples drawn from a distribution different from the one upon which it was trained. We therefore assume there are two different distributions, $\mathcal{D}^{source}$ and $\mathcal{D}^{target}$, from which data may be drawn. Given this notation we can then precisely state the transfer learning problem as trying to assign labels Y_{test}^{target} to test data X_{test}^{target} drawn from $\mathcal{D}^{target}$, given training data $(X_{train}^{source}, Y_{train}^{source})$ drawn from $\mathcal{D}^{source}$.

In this thesis we focus on two subproblems of transfer learning:

- *domain adaptation*, where we assume Y (the set of possible labels) is the same for both D^{source} and D^{target}, while D^{source} and D^{target} themselves are allowed to vary between domains.
- *multi-task learning* [Ando and Zhang, 2005; Caruana, 1997; Sutton and McCallum, 2005; Zhang et al., 2005] in which the task (and label set) is allowed to vary from source to target.

Domain adaptation can be further distinguished by the degree of relatedness between the source and target domains. For example, in this work we group data collected in the same medium (e.g., all annotated e-mails or all annotated news articles) as belonging to the same

genre. Although the specific boundary between domain and genre for a particular set of data is often subjective, it is nevertheless a useful distinction to draw.

One common way of addressing the transfer learning problem is to use a *prior* which, in conjunction with a probabilistic model, allows one to specify *a priori* beliefs about a distribution, thus biasing the results a learning algorithm would have produced had it only been allowed to see the training data [Raina et al., 2006]. In the example from §1.1, our belief that capitalization is less strict in instant messages than in encyclopedia articles could be encoded in a prior that biased the importance of the `capitalization` feature to be lower for instant messages than encyclopedia articles. In Section 3.1 we address the problem of how to come up with a suitable prior for transfer learning across named entity recognition problems.

2.1.2 Domain adaptation

Domain adaptation is distinct from other forms of transfer learning (such as multitask learning [Ando and Zhang, 2005; Caruana, 1997; Sutton and McCallum, 2005; Zhang et al., 2005]) because we are assuming that the set of possible labels, Y, remains constant across the various domains, while allowing the distribution of X and, most importantly, $Y|X$ to change. In our setting, the labels, Y, are members of the binary set {PROTEIN, NON_PROTEIN}, while the instances, X, are the tokens of the documents themselves. Another important example of domain adaptation is *concept drift*, in which the source and target data's distributions start out identical, but drift farther and farther apart from each other over time [Widmer and Kubat, 1996].

In prior work, different researchers have made different assumptions about the relationship between the source and target domain, a defining characteristic of domain adaptation. In the supervised setting, one can directly compare both the marginal and conditional distributions

of the data in both domains, looking for patterns of generalizability across domains [Daumé III and Marcu, 2006; Jiang and Zhai, 2006; Daumé III, 2007], as well as examining the common structure of related problems [Ben-David et al., 2007; Schölkopf et al., 2005; Arnold et al., 2008; Blei et al., 2002]. There is likewise work that tries to quantify these inter-domain relationships in the unsupervised [Arnold et al., 2007], semi-supervised [Grandvalet and Bengio, 2005; Blitzer et al., 2006], and transductive learning settings [Taskar et al., 2003]. Similarly, in the biological domain, there has been work on using semi-supervised machine learning techniques to extract protein names by combining dictionaries with large, full-text corpora [Shi and Campagne, 2005], but without the explicit modeling of differences between data domains that we attempt in this thesis. In our work, we take advantage of the fact that the source and target domains are different sections of the same structured document and use this fact to develop features that are robust across those different domains.

2.1.3 Multi-task learning

Whereas in domain adaptation the set of possible labels for our learning task, Y, is held constant between source and target data, in the *multi-task* setting this label set, or *task*, is allowed to vary between the source task and target task [Ando and Zhang, 2005; Caruana, 1997; Sutton and McCallum, 2005; Zhang et al., 2005; Ghamrawi and McCallum, 2005]. Expanding on the example from Section 1.1, this would be like using encyclopedia articles labeled with personal names in order to train an extractor to find place names in those same types of articles. Again, there is an obvious overlap between these two learning problems and the goal of multi-task learning is to investigate how best to characterize and exploit this similarity. More nefariously, not only are the labels themselves allowed to change, but also the intended semantics of those labels. For example, the two semantically distinct problems of labeling tokens as people or places can both be represented by the same binary labeling

scheme.

Although there seems to be a clear formal distinction between domain adaptation and multi-task learning, in this work we tend to consider them in much the same way. Our thesis's goal is to find robust ways of learning using as many different sources of data as we have available. Just as the data we use can come from many related domains, so too our labels (where they are available) are allowed to refer to a number of distinct, though inter-related tasks. Thus, for much of this thesis we will use the term 'task' (or alternately, *setting*) to refer both to the distribution from which our training and test data are drawn and the set of labels which our learning is trying to predict.

2.1.4 Semi-supervised learning

Analogously to multi-task learning, where we try to make use of data with labels related to our source task, in the semi-supervised setting we try to make use of data with no labels at all [Abney, 2007; Collins and Singer, 1999; Yarowsky, 1995]. Indeed, in the multi-task framework, any data for which *all* labels for *all* tasks are not available can be considered, in some sense, semi-supervised. In this way, as presented in Figure 2.1, we consider semi-supervised learning an extra dimension of the robust learning framework that one can combine with an existing technique by making use of what unlabeled data is available. In the supervised setting, the data is usually segmented into two disjoint subsets: the training set $(X_{train}, Y_{train}) = \{(x_1, y_1) \cdots, (x_N, y_N)\}$, which can be used for training, and the test set $X_{test} = (x_1, \cdots, x_M)$, for which labels are not available at training time. In the semi-supervised setting [Zhu, 2005], the training data is supplemented with a set of **auxiliary** data, $X_{aux} = (x_1, \cdots, x_P)$, for which no corresponding labels are provided. When using semi-supervised techniques for transfer learning, the distribution from which this unlabeled auxiliary data is drawn is allowed to vary.

2.1.5 Non-transfer robustness

Despite recent interest in and research into the problems of transfer learning as such, the idea of robust learning itself is not a new one. Feature selection has proved a very effective means of generating robust learners, especially when regularized for sparsity, as in the case of lasso and least angles regression [Tibshirani, 1996; Efron et al., 2004]; or when the features are designed to succinctly summarize the relevant information contained in a dataset, as in principal components analysis [Jolliffe, 2002] and mutual information techniques [Zaffalon and Hutter, 2002]; or when they are engineered to be resilient to deletion [Globerson and Roweis, 2006]. Researchers have also tried engineering and selecting features themselves that they believe will be robust to noise and shifts in the data [Janche and Abney, 2002]. Relatedly, a whole range of expectation maximization (EM) techniques have been developed for learning in situations where not all relevant information is available [Dempster et al., 1977; Ghahramami and Jordan, 1994]. In this thesis we build on many these techniques, combining and extending them where necessary.

2.2 Examples of transfer learning settings & techniques

In the first two sections below (§2.2.1-2.2.2), we introduce and discuss several examples of learning across the spectrum of transfer problems [Arnold et al., 2007]. These problems vary with respect to what labels and data are available from the source and target domains at train time. They are also summarized in Table 2.1 for the reader's convenience. Later, we survey some popular approaches to these types of problems (§2.2.3-2.2.5), and then present some comparative results to make the algorithms' relative strengths and weaknesses more concrete (§2.2.6, Table 2.4).

Table 2.1: Learning settings are summarized by the type of auxiliary and test data used. For all settings we assume $(X_{train}^{source}, Y_{train}^{source})$ is available at training time, while Y_{test} is unknown. Settings for which we have run experiments are marked in bold (c.f. Table 2.4). Some settings are omitted where they do not correspond to a known natural example.

Natural name for learning setting	Auxiliary data		Test data	
	Domain	Labels	Domain	X_{test}
Inductive learning	-	-	$\mathcal{D}^{source}$	unseen
Semi-supervised inductive learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	unseen
Transductive learning	-	-	$\mathcal{D}^{source}$	seen
Transfer learning	-	-	$\mathcal{D}^{target}$	unseen
Inductive transfer learning	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	unseen
Semi-supervised inductive transfer learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	unseen
Transductive transfer learning	-	-	$\mathcal{D}^{target}$	seen
Supervised Transductive transfer learning	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	seen
Relaxed Transductive transfer learning[1]	-	-	$\mathcal{D}^{target}$	seen
Semi-supervised transductive transfer learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	seen

[1] A relaxation of transductive transfer learning in which proportions of labels in the target data is known at training time.

2.2.1 Inductive learning

In the paradigm of *inductive learning*, (X_{train}, Y_{train}) are known, while both X_{test} and Y_{test} are completely hidden during training time. In the case of *semi-supervised* inductive learning [Zhu, 2005; Sindhwani et al., 2005; Grandvalet and Bengio, 2005], the learner is also provided with auxiliary unlabeled data $X_{auxiliary}$, that is not part of the test set. It has been noted that such auxiliary data typically helps boost the performance of the classifier significantly.

2.2.2 Transductive learning

Another setting that is closely related to semi-supervised learning is *transductive learning* [Vapnik, 1998; Joachims, 1999; Joachims, 2003], in which X_{test} (but, importantly, not Y_{test}), is known at training time. That is, the learning algorithm knows exactly which examples it will be evaluated on after training. This can be a great asset to the algorithm, allowing it to shape its decision function to match and exploit the properties seen in X_{test}. One can think of transductive learning as a special case of semi-supervised learning in which $X_{auxiliary} = X_{test}$.

In the three cases discussed above, X_{test} and X_{train} are both assumed to have been drawn from the same distribution, $\mathcal{D}$. As mentioned previously, however, we are more interested in the case where these distributions are allowed to differ, that is, the transfer learning setting. One of the first formulations of the transfer learning problem was presented over 10 years ago by Thrun [Thrun, 1996]. More recently there has been a focus on using source data to learn various types of priors for the target data [Raina et al., 2006]. Other techniques have tried to quantify the generalizability of certain features across domains [Daumé III and Marcu, 2006; Jiang and Zhai, 2006], or tried to exploit the common structure of related

problems [Ben-David et al., 2007; Blitzer et al., 2006].

Although the case of transfer learning without access to any data drawn from $\mathcal{D}^{target}$ is not completely hopeless [Jiang and Zhai, 2006], in this thesis we choose to focus on extensions to the transfer learning setting that allow us to capture some information about D^{target}. One obvious such setting is *inductive transfer learning* where we also provide a few auxiliary labeled data $(X^{target}_{auxiliary}, Y^{target}_{auxiliary})$ from the target domain in addition to the labeled data from the source domain. Due to the presence of labeled target data, this method could also be called *supervised transfer* learning and is the most common setting used by researchers in transfer learning today.

There has also been work on *transductive transfer learning*, where there is no auxiliary labeled data in the target domain available for training, but where the unlabeled test set on the target domain X^{target}_{test} can be seen during training. Again, due to the lack of labeled target data, this setting could be considered *unsupervised transfer* learning. It is important to point out that *transductive learning* is orthogonal to *transfer learning*. That is, one can have a transductive algorithm that does or does not make the transfer learning assumption, and vice versa. Much of the work in this thesis is inspired by the belief that, although distinct, these problems are nevertheless intimately related. More specifically, when trying to solve a transfer problem between two domains or tasks, it seems intuitive that looking at the possibly *unlabeled* data of the target domain, or another related task, during training will improve performance over ignoring this source of information.

We note that the setting of *inductive transfer learning*, in which labeled data from both source and target domains are available for training, serves as an upper-bound to the performance of a learner based on *transductive transfer learning*, in which no labeled target data is available. For similar reasons, we considered an additional artificial setting, which we call *relaxed transductive transfer learning*, in our experiments. This setting is almost equivalent to

the transductive transfer setting, but the model is allowed to know the proportion of positive examples in the target domain. Although this technically violates the terms of unsupervision in transductive transfer learning, in practice estimating this single parameter over the target domain does not require nearly as much labeled target data as learning all the parameters of a fully supervised transfer model, and thus serves as a nice compromise between the two extremes of transduction and supervision. Practically, this proportion is useful to know for determining thresholds [Yang, 2001] and guaranteeing certain semi-supervised performance results [Blum and Mitchell, 1998].

These and a few other interesting settings are summarized in Table 2.1. Note that we only displayed a small subset of the many possible learning settings.

2.2.3 Naive Bayes classifier

Inductive learning: maximum likelihood estimation

Naive Bayes [McCallum and Nigam, 1998] is one of the most popular and effective generative classifiers for many text-classification tasks. Like any generative model, its decision rule is given by the posterior probability of the class y given the example x, given by $P(y|x)$, which is computed using Bayes' rule as follows:

$$P(y|x) = \frac{P(x|\theta(y))\pi(y)}{\sum_{y'} P(x|\theta(y'))\pi(y')} \tag{2.1}$$

where $\theta(y)$ are the class-conditional parameters and $\pi(y)$ are the prior probabilities. The naive Bayes model makes the somewhat unrealistic yet practical assumption of conditional-independence between the features of each example, given its class. That is:

$$P(x|\theta(y)) = \prod_{j=1}^{F} P(f_j(x)|\theta_j(y)) \tag{2.2}$$

In our case, since the features are all binary, we use the Bernoulli distribution to model each

feature as follows:

$$P(x|\theta(y)) = \prod_{j=1}^{F} (\theta_j(y))^{f_j(x)} (1 - \theta_j(y))^{1-f_j(x)} \tag{2.3}$$

where $\theta_j(y)$ can be interpreted as the probability that the feature f_j assumes a value 1 given the class y. The Bernoulli parameters $\theta_j(y)$ and $\pi(y)$ are estimated using Maximum Likelihood training with the labeled training data $(X_{train}, Y_{train}) = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ as below:

$$\begin{aligned} \theta_j(y) &= \frac{\sum_{i=1}^{N} f_j(x_i)\delta_y(y_i) + \lambda}{\sum_{i=1}^{N} \delta_y(y_i) + 2\lambda} \\ \pi(y) &= \frac{\sum_{i=1}^{N} \delta_y(y_i)}{N} \end{aligned} \tag{2.4}$$

where $\delta_y(y_i) = 1$ if $y = y_i$ and 0 otherwise; and λ is the Laplace smoothing parameter, which we set to 0.05 in our experiments.

Inductive transfer learning: maximum likelihood estimation with concatenated data

In the *inductive transfer* case we concatenate the entire labeled data $(X_{train}^{source}, Y_{train}^{source})$ and $(X_{train}^{target}, Y_{train}^{target})$ to generate a single training set. Then, we learn the parameters $\theta_j(y)$ and $\pi(y)$ using the maximum likelihood estimators shown in the classic supervised case (see eqn. 2.4). Although more sophisticated approaches are possible, we tried this algorithm as a simple baseline.

Transductive transfer learning: source-initialized EM

In the transductive transfer case, $(X_{train}^{target}, Y_{train}^{target})$ are not available for training, but X_{test}^{target} is available at training time. Learning from unlabeled examples in the generative framework is done typically using the standard Expectation Maximization algorithm [Nigam et al., 2000]. The algorithm is iterative, and consists of two steps: in the E-step corresponding to the t^{th} iteration, we compute the posterior probability of each label for all the unlabeled examples

w.r.t. the old parameter values $\theta_j^{(t)}(y), \pi^{(t)}(y)$ as follows:

$$\forall_y P(y|x, \theta^{(t)}, \pi^{(t)}) = \frac{P(x|\theta^{(t)}(y))\pi^{(t)}(y)}{\sum_{y'} P(x|\theta^{(t)}(y'))\pi^{(t)}(y')} \tag{2.5}$$

In the M-step, we estimate the new parameters $\theta_j^{(t+1)}(y), \pi^{(t+1)}(y)$ using the posterior probabilities as follows.

$$\theta_j^{(t+1)}(y) = \frac{\sum_{i=1}^{N} f_j(x_i) P(y|x_i, \theta_j^{(t)}(y))}{\sum_{i=1}^{N} P(y|x_i, \theta_j^{(t)}(y))} \tag{2.6}$$

$$\pi^{(t+1)}(y) = \frac{\sum_{i=1}^{N} P(y|x_i, \theta_j^{(t)}(y))}{N} \tag{2.7}$$

where N is the number of unlabeled examples available during training. In our case, this is the size of the set X_{test}^{target}. The iterations are continued until the likelihood of the unlabeled data converges to a maximum value. In the completely unsupervised case of the EM algorithm, the model parameters are initialized to random values before starting the iterations. In our case, since we have $(X_{train}^{source}, Y_{train}^{source})$ at our disposal, we first do a classic supervised training of our model using the labeled source data, and initialize the parameters to the ones learned from the source data, before we start the EM iterations. This encodes the information available from the source data into the model, while allowing the EM algorithm to discover its optimal parameters on the target domain.

Relaxed transductive transfer learning: redefining the prior

In the case when the values of the prior probability of each class in the target data is available, we simply fix $\pi(y)$ to these values and only estimate $\theta(y)$ using eqn. 2.6 in the M-step of the EM algorithm.

2.2.4 Maximum entropy

Entropy maximization (MaxEnt) [Berger et al., 1996; Nigam et al., 1999] is a way of modeling the conditional distribution of labels given examples. Given a set of training examples $X_{train} \equiv \{x_1, \ldots, x_N\}$, their labels $Y_{train} \equiv \{y_1, \ldots, y_N\}$, and the set of features $\mathcal{F} \equiv \{f_1, \ldots, f_F\}$, MaxEnt learns a model consisting of a set of weights corresponding to each class $\Lambda = \{\lambda_{1,y} \ldots \lambda_{F,y}\}_{y \in \{0,1\}}$ over the features so as to maximize the conditional likelihood of the training data, $\mathrm{p}(Y_{train}|X_{train})$, given the model p_Λ. In exponential parametric form, this conditional likelihood can be expressed as:

$$p_\Lambda(y_i = y|x_i) = \frac{1}{Z(x_i)} \exp(\sum_{j=1}^{F} f_j(x_i)\lambda_{j,y}) \tag{2.8}$$

where Z is the normalization term.

In order to avoid overfitting the training data, these λ's are often further constrained by the use of a Gaussian prior [Chen and Rosenfeld, 1999] with diagonal covariance, $\mathcal{N}(\mu, \sigma^2)$, which tries to maximize:

$$\sum_{j,y} log \frac{1}{\sqrt{2\pi\sigma_{j,y}^2}} exp(-\frac{(\lambda_{j,y} - \mu_{j,y})^2}{2\sigma_{j,y}^2}) \tag{2.9}$$

Thus the entire expression being optimized is:

$$\underset{\Lambda}{\operatorname{argmax}} \sum_{i=1}^{N} \left(\log\ p_\Lambda(y_i|x_i) - \beta \sum_{j}^{F} \frac{(\lambda_{j,i} - \mu_{j,i})^2}{\sigma_{j,i}^2} \right) \tag{2.10}$$

where $\beta > 0$ is a parameter controlling the amount of regularization. Maximizing this likelihood is equivalent to constraining the joint expectations of each feature and label in the learned model, $E_\Lambda[f_j, y]$, to match the Gaussian-smoothed empirical expectations $E_{train}[f_j, y]$ as shown below:

$$E_{train}\left[f_j, y\right] = \frac{1}{N} \sum_{i}^{N} \left(f_j(x_i)\delta_y(y_i) - \frac{\lambda_{j,i} - \mu_{j,i}}{\sigma_{j,i}^2} \right) \tag{2.11}$$

$$E_\Lambda\left[f_j, y\right] = \frac{1}{N} \sum_{i}^{N} f_j(x_i) P_\Lambda(y|x_i) \tag{2.12}$$

where $\delta_y(y_i) = 1$ if $y = y_i$ and 0 otherwise. In the next few subsections, we will describe how we adapt the model to various scenarios of transfer learning.

Conditional random fields (instance structure)

When it comes to actually training a model, we need a learning algorithm that can integrate and balance the variety of features and disparate sources of information we are trying to exploit. We used **conditional random fields** (CRF's) [Lafferty et al., 2001], a generalization of the common maximum entropy model from the i.i.d. case (where each token is classified in isolation), to the sequential case (where each token's classification influences the classification of its neighbors). This attribute is especially useful in a setting such as domain adaptation, where we would like to spread high-confidence predictions made on examples resembling the source domain to lower-confidence predictions of less familiar target domain instances. Similarly, like maximum entropy models, CRF's allow great flexibility with respect to the definition of the model's features, freeing us from worrying about the relative independence of specific features, while maintaining the crucial focus on the locality of features.

The parametric form of the CRF for a sentence of length n is given as follows:

$$p_\Lambda(\mathbf{Y} = \mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(\sum_{i=1}^{n} \sum_{j=1}^{F} f_j(\mathbf{x}, y_i)\lambda_j) \tag{2.13}$$

where $Z(\mathbf{x})$ is the normalization term. CRF learns a model consisting of a set of weights $\Lambda = \{\lambda_1 ... \lambda_F\}$ over the features so as to maximize the conditional likelihood of the training data, $\mathrm{p}(Y_{train}|X_{train})$, given the model p_Λ.

CRF with Gaussian priors

To avoid overfitting the training data, these λ's are often further constrained by the use of a Gaussian prior [Chen and Rosenfeld, 1999] with diagonal covariance, $\mathcal{N}(\mu, \sigma^2)$, which tries

to maximize:

$$\underset{\Lambda}{\operatorname{argmax}} \sum_{k=1}^{N} \left(\log \ p_\Lambda(\mathbf{y}_k|\mathbf{x}_k) \right) - \beta \sum_{j}^{F} \frac{(\lambda_j - \mu_j)^2}{2\sigma_j^2}$$

where $\beta > 0$ is a parameter controlling the amount of regularization, and N is the number of sentences in the training set.

Inductive transfer: Source trained prior models (*Chelba-Acero*)

One recently proposed method [Chelba and Acero, 2004] for transfer learning in MaxEnt models, which we call the *Chelba-Acero* model. involves modifying Λ's regularization term. First a model of the source domain, Λ^{source}, is learned by training on $\{X_{train}^{source}, Y_{train}^{source}\}$. Then a model of the target domain is trained over a limited set of labeled target data $\left\{X_{train}^{target}, Y_{train}^{target}\right\}$, but instead of regularizing this Λ^{target} to be near zero by minimizing $\|\Lambda^{target}\|_2^2$, Λ^{target} is instead regularized towards the previously learned source values Λ^{source} by minimizing $\|\Lambda^{target} - \Lambda^{source}\|_2^2$. Thus the modified optimization problem is:

$$\underset{\Lambda^{target}}{\operatorname{argmax}} \sum_{i=1}^{N_{train}^{target}} \log \ p_{\Lambda^{target}}(y_i|x_i) - \beta\|\Lambda^{target} - \Lambda^{source}\|_2^2 \tag{2.14}$$

where N_{train}^{target} is the number of labeled training examples in the target domain. It should be noted that this model requires Y_{train}^{target} in order to learn Λ^{target} and is therefore a supervised form of *inductive transfer*.

Feature space expansion (*Daumé)*

Another approach to the problem of inductive transfer learning is explored by Daumé [Daumé III, 2007; Daumé III and Marcu, 2006]. Here the idea is that there are certain features that are common between different domains, and others that are particular to one or the other. More specifically, we can redefine our feature set $\mathcal{F}$ as being composed of two distinct subsets $\mathcal{F}^{specific} \bigcup \mathcal{F}^{general}$, where the conditional distribution of the features in $\mathcal{F}^{specific}$

differ between X^{source} and X^{target}, while the features in $\mathcal{F}^{general}$ are identically distributed in the source and target. Given this assumption, there is an EM-like algorithm [Daumé III and Marcu, 2006] for estimating the parameters of these distributions. There is also a simpler approach [Daumé III, 2007] of just making a duplicate copy of each feature in X^{source} and X^{target}, so whereas before you had $x_i = \langle f_1(x_i)...f_F(x_i)\rangle$, you now have

$$\begin{aligned} x_i = \langle\ & f_1(x_i)^{specific}, f_1(x_i)^{general} \\ & ...f_F(x_i)^{specific}, f_F(x_i)^{general}\ \rangle \end{aligned} \tag{2.15}$$

where $specific$ is *source* or *target* respectively, and $f_j(x_i)^{specific}$ is just a copy of $f_j(x_i)^{general}$. The idea is that by expanding the feature space in this way MaxEnt (or any other learner) will be able to assign different weights to different versions of the same feature. If a feature is common in both domains its *general* copy will get most of the weight, while its specific copies (f^{source} and f^{target}) will get less weight, and vice versa.

Transductive transfer learning

Transductive learning under the MaxEnt framework can be performed analogously to the naive Bayes method. Similarly, given a prior estimate of the probability of each class label in the test data, relaxed transductive learning can also be performed.

2.2.5 Support vector machines (SVM)

Support vector machines (SVM's) [Joachims, 2002] take a different approach to the binary classification problem. Instead of explicitly modeling the conditional distribution of the data and using these estimates to predict labels, SVMs try to model the data geometrically. Each example is represented as an F-dimensional real-valued vector of features and is then projected as a point in F-dimensional space.

The *inductive SVM* exploits the label information of the training data and fits a discrim-

inative hyperplane between the positively and negatively labeled training examples in this space, so as to best separate the two classes. This separation is called the margin, and thus SVMs belong to the margin based approach to classification. This formulation has proven very successful as inductive SVMs currently have some of the best general performance of any popular machine learning algorithm.

Inductive SVM

Recall that in the supervised inductive transfer case, we are given the training sets $(X_{train}^{source}, Y_{train}^{source})$ and $(X_{train}^{target}, Y_{train}^{target})$. Since the SVM does not explicitly model the data distribution, we simply concatenate the source and target labeled data together and provide the entire data for training. The hope is that it will improve on an SVM trained purely on labeled source data, by re-adjusting its hyperplane based on the labeled target data. It is possible to do better than such a naive approach [1], but we used this as a reasonable baseline.

Transductive SVM

Transduction with SVMs, due to their geometric interpretation, is quite intuitive. Whereas, in the supervised case, we tried to fit a hyperplane to best separate the labeled training data, in the transductive case, we add in unlabeled testing data which we must also separate. Since we do not know the labels of the testing data, however, we cannot perform a straight forward margin maximization, as in the supervised case. Instead, one can use an iterative algorithm [Joachims, 1999]. Specifically, a hyperplane is trained on the labeled source data and then used to classify the unlabeled testing data. One can adjust how confident the hyperplane must be in its prediction in order to use a pseudo-label during the next phase of training (since there are no probabilities, large margin values are used as a measure of confidence). The pseudo-labeled testing data is then, in turn, incorporated in the next

[1] For example, one could impose a higher penalty for classification errors on the target data than on the source data.

round of training. The idea is to iteratively adjust the hyperplane (by switching presumed pseudo-labels) until it is very confident on most of the testing points, while still performing well on the labeled training points.

Transductive SVMs were originally designed for the case where the training and test sets were drawn from the same domain. Again, since SVMs do not model the data distribution, it is not immediately obvious how one would model different distributions in the SVM algorithm. Hence in this work, we directly test the applicability of transductive SVMs to the transductive transfer setting.

Relaxed transductive SVM: tweaking the margin

Just as in the probabilistic naive Bayes and MaxEnt settings prior knowledge of class proportions in the test data could be leveraged to improve cross domain learning by adjusting the prior probability of each class label, similarly in the SVM setting this same information can be used to adjust the margin and penalty assessed for each misclassified training example of each class. For instance, if one expects more positive examples in the test data, then to train a learner that minimizes expected performance over the test data, one should penalize errors on positive training data (false negatives) more severeley than errors on negative training data (false positives), since these will occur more often in the test data.

2.2.6 Comparison of existing techniques

Domain

We now turn to *protein name extraction*, an interesting problem domain [Shi and Campagne, 2005; Wang et al., 2008; Ji et al., 2002] in which to compare these methods within various learning settings. In this problem you are given text related to biological research (usually

Table 2.2: Summary of data used in experiments

Corpus name (Abbr.)	Abstracts	Tokens	% Positive
UTexas (UT)	748	216,795	6.6%
Yapex (Y)	200	60,530	15.0%
Yapex-train (YTR)	160	48,417	15.1%
Yapex-test (YTT)	40	12,113	14.5%

abstracts, captions, and full body text from biological journal articles) which is known to contain mentions of protein names. The goal is to identify which words are part of a protein name mention, and which are not. One major difficulty is that there is a large variance in how these proteins are mentioned and annotated between different authors, journals, and sub-disciplines of biology. Because of this variance it is often difficult to collect a large corpus of truly identically distributed training examples. Instead, researchers are often faced with heterogeneous sources of data, both for training and testing, thus violating one of the key assumptions of most standard machine learning algorithms. Hence the setting of transfer learning is very relevant and appropriate to this problem.

Data and evaluation

Our corpora are abstracts from biological journals coming from two sources: University of Texas, Austin (UT) [Bunescu et al., 2004] and Yapex [Franzén et al., 2002]. Each abstract was tokenized and each token was hand-labeled as either being part of a protein name or not. We used a standard natural language toolkit [Cohen, 2004] to compute tens of thousands of binary features on each of these tokens, encoding such information as capitalization patterns and contextual information of surrounding words.

Some summary statistics for these data are shown in Table 2.2. We purposely chose corpora that differed in two important dimensions: the total amount of data collected and the relative

Table 2.3: Training and testing data used in the settings of Inductive learning (I), Inductive Transfer (IT), Transductive Transfer (TT) and Relaxed Transductive Transfer (RTT). Abbreviations of data sets are described in Table 2.2.

Setting	Source-train	Target-train	Target-test
I	-	YTR	YTT
IT	UT	YTR	YTT
TT	UT	-	Y
RTT	UT	-	Y

proportion of positively labeled examples in each dataset. Specifically, UT has over three times as many tokens as Yapex but has only half the proportion of positively labeled protein names. This disparity is not uncommon in the domain and could be attributed to differing ways the data sources were collected and annotated. Specifically, if the protein mention annotations in Yapex tend to be longer (that is, extend for more tokens) then the proportion of positively labeled tokens will be higher in Yapex. For all our experiments, we used the larger UT dataset as our source domain and the smaller Yapex dataset as our target. We also split the Yapex data into two parts: *Yapex-train* (YTR) consisting of 80% of the data, and *Yapex-test* (YTT), consisting of the remaining 20%.

In Table 2.3, we display the subsets of data used for various learning settings in our experiments. Note that the transductive methods use different testing data from the inductive methods. This choice is made deliberately to provide a chance for the classifiers in each setting to achieve their peak performance, i.e., transductive algorithms work best when there is abundance of unlabeled test data and inductive algorithms work best when there is plenty of labeled data. However, since the data is slightly different between inductive and transductive settings, one must use caution in comparing the transductive results to the inductive ones.

Because of the relatively small proportion of positive examples in both the UT and Yapex datasets, we are more interested in achieving both high precision and recall of protein name mentions instead of simply maximizing classification accuracy. Since we were dealing with binary, and not sequential classification, the definition of these measures is straightforward as summarized below:

$$\begin{aligned} \text{accuracy} &= \frac{\text{\# of tokens labeled correctly by the model}}{\text{total \# of tokens}} \\ \text{precision} &= \frac{\text{\# of POS-tokens labeled POS by the model}}{\text{\# of tokens labeled POS by the model}} \\ \text{recall} &= \frac{\text{\# of POS-tokens labeled POS by the model}}{\text{\# of POS-tokens}} \\ \text{F1} &= \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \end{aligned} \tag{2.16}$$

We use the $F1$ measure, which combines precision and recall into one metric, as our main evaluation measure. These metrics are evaluated on the level of tokens, as opposed to multi-token spans, since this provides a simple binary distinction that is a nice test case for comparison to other machine learning studies, and avoids any complications of ambiguous or noisy span boundaries.

Experiments and results

We assessed the relative performance of these methods on the four different learning settings described in previous sections. We restricted ourselves to a limited evaluation since the goal of these experiments was to concretely illustrate the various learning settings, rather than provide an exhaustive comparison of methods.

In addition to running the corresponding adaptations of each model for each of the settings, we did a few additional runs across the settings for purposes of illustration. For example, we ran the transductive SVM not only on the transductive settings, but also on the two inductive settings. Note that TSVM, when run on the inductive case corresponds to transductive

Table 2.4: Summary of % precision (**Prec**), recall (**Rec**), and F1 for regular maximum entropy (**Basic**), prior-based regularized MaxEnt (**Chelba-Acero**), and feature expansion MaxEnt (**Daumé**), inductive SVM (**ISVM**), transductive SVM (**TSVM**), Maximum Likelihood Naive Bayes (**NB-ML**), and EM based Naive Bayes (**NB-EM**) models under the conditions of classic inductive learning, (**Induction**), unsupervised transductive transfer learning, (**TransductTransfer**), relaxed transductive transfer, (**RelaxTransductTransfer**), and supervised inductive transfer (**InductTransfer**), as introduced in the previous sections and summarized in Table 2.1. **F1** measures are presented in **bold**.

Method	Induction			TransductTransfer			RelaxTransductTransfer			InductTransfer		
	Prec	Rec	**F1**	Prec	Rec	**F1**	Prec	Rec	**F1**	Prec	Rec	**F1**
MAXIMUM ENTROPY												
Basic	85	78	**82**	75	42	**54**	65	68	**67**	81	54	**65**
Chelba-Acero	-	-	-	-	-	-	-	-	-	87	84	**85**
Daumé	-	-	-	-	-	-	-	-	-	84	62	**72**
SUPPORT VECTOR MACHINES												
ISVM	78	58	**67**	86	40	**54**	86	40	**55**	86	52	**65**
TSVM	68	79	**73**	86	46	**60**	72	75	**73**	86	58	**70**
NAIVE BAYES												
NB-ML	80	93	**86**	50	81	**62**	48	85	**61**	55	84	**67**
NB-EM	-	-	-	40	84	**54**	41	82	**55**	-	-	-

learning (see Table 2.1) and when run on the inductive transfer case, corresponds to the supervised transductive transfer learning in Table 2.1. There are other extra runs we did for the purposes of comparison, which will become apparent from the following discussion.

Table 2.4 summarizes the results under all four settings. The inductive experiment is dominated by Naive Bayes, achieving an F1 of 86% compared to MaxEnt's 82% and TSVM's 73%. This should not be surprising since generative models are known to be robust when a large amount of labeled training data is available.

Moving to the transductive transfer setting causes all three methods' performances to fall, but MaxEnt falls most sharply, causing it to lose its entire lead over TSVM. Note that in this setting, basic MaxEnt and ISVM have equivalent performance of about 54% F1. The inductive Naive Bayes (using maximum likelihood estimator) proves to the top performer in this setting. TSVM, on the other-hand, is able to adjust its hyperplane in light of the transfer test data and stabilize its performance at 60%, even though it is unlabeled, because it knows where these points lie relative to the labeled training points in feature space. The transductive version of the naive Bayes (using EM), however, fares worse than its inductive counterpart. Since EM's optimization function is the marginal log-likelihood of the test data, without knowledge of the test's conditional distribution, it is not guaranteed to improve the classification performance in some cases.

In the relaxed transductive transfer setting, finally, where the target dataset is still unlabeled but all algorithms are told the expected proportion of positive examples, TSVM excels. Again, while MaxEnt is able to make significant use of this information (note the jump to 67% from 54%), it seems TSVM does a better job leveraging the prior knowledge into better performance. Maximum Likelihood based Naive Bayes, on the other hand loses out. It seems that the class conditional probability is more critical in naive Bayes than the prior, so tuning the latter's value does not have any positive impact on its performance. Also, notice that

the EM based naive Bayes is even worse, repeating the pattern in the transductive transfer case.

Finally, the last column of Table 2.4 compares the performance of the three methods for inductive transfer learning: the prior-based regularized maximum entropy method (*Chelba-Acero*, described in section 2.2.4), and the feature expanding version (*Daumé*, described in section 2.2.4). We can see that both methods handily outperform the transductive transfer methods described in the second column of Table 2.4, and for the most part outperform even the relaxed transductive transfer versions in column three. This should not be surprising given the fact that the inductive transfer methods can actually see some labeled examples from the target domain and thus, in the case of MaxEnt, better estimate the conditional expectation of the features in the target data. Likewise, since they have access to labeled target data, they can also assess the proportion of positive examples and adjust their decision functions accordingly. What is more surprising, however, is the fact that these methods do not significantly outperform the inductive learning methods described in the first column of Table 2.4. This suggests that these inductive transfer methods are relying almost entirely on their labeled target data in order to train their classifiers, and are not making full use of the large amount of labeled source data. One might assume that having access to almost four times as much related data, in the form of the labeled source data, would significantly boost their ability to classify the target data (this is, after all, one of the stated goals of transfer learning). Dishearteningly, in this instance, this seems not to be the case. The regularized maximum entropy model *Chelba-Acero* does outperform[2] the basic MaxEnt in the inductive setting, but not by as much as might have been hoped for.

In order to measure how much these inductive transfer methods' explicit modeling of the transfer problem was responsible for their performance, we compared them to the baselines

[2] *Chelba-Acero* has F1 of 85 vs. *MaxEnt*'s 82. Significance was determined by comparing the 99% binomial confidence intervals for each method's recall and precision.

of ISVM, TSVM, MaxEnt and Naive Bayes trained on a simple concatenation of the labeled source and target training data. These transfer-agnostic methods clearly benefited from the addition of labeled target data (as compared to column *TransductiveTransfer*), yet still yielded consistently lower F1 than the transfer-aware *Chelba-Acero* and *Daumé* methods, suggesting that the mere presence of labeled sets of both types (source and target) of data is not enough to account for the transfer methods' superior results. Instead, it seems it is the modeling of the different domains in the transfer problem, even in simple ways, that provides the extra boost to performance.

Conclusions

These experiments and analysis have shed light on a number of important issues and considerations related to the problems of transduction and transfer learning.

We have seen that in the case of discriminative models, even a small amount of prior knowledge about the target domain can greatly improve performance in a transductive transfer problem. The generative model is not able to exploit this information. For all these models, we notice that even large amounts of source data cannot overcome the advantage of having access to labeled data drawn from the target distribution.

We have also seen the degree to which pseudo-labeling based schemes can improve performance by incorporating the unlabeled structure of the target domain. However, this improvement is not seen in the generative Naive Bayes model. We believe this is because discriminative models directly optimize classification accuracy, while the EM based Naive Bayes model optimizes an unrelated function, namely, the marginal log-likelihood.

Finally, we have seen that the generative Naive Bayes model is robust in the inductive setting with large amount of labeled data, while the discriminative models are at least as good or better in the transductive setting. Of the two discriminative models considered, the margin based SVM seems to adapt better to the unlabeled data.

These insights regarding the benefits of prior domain knowledge, pseudo labels, and labeled target data will be leveraged again in Sections 3, 6.2 and 5 respectively to create our own robust NER learners.

Chapter 3

Hierarchical Feature Models

In this chapter we draw on the results of the previous section indicating the utility of domain-specific priors, and develop a lexically-motivated hierarchical model of our domain's feature space that can be used to construct robust priors for domain-adaptive named entity recognition.

3.1 Definition of hierarchical feature models

By exploiting the hierarchical relationship present in many different natural language feature spaces, we are able to transfer knowledge across domains, both relating similar features to one another, while allowing distinct ones to vary across domains, genres and tasks [Arnold et al., 2008].

3.1.1 Hierarchical feature trees

In many NER problems, features are often constructed as a series of transformations of the input training data, performed in sequence. Thus, if our task is to identify tokens as either

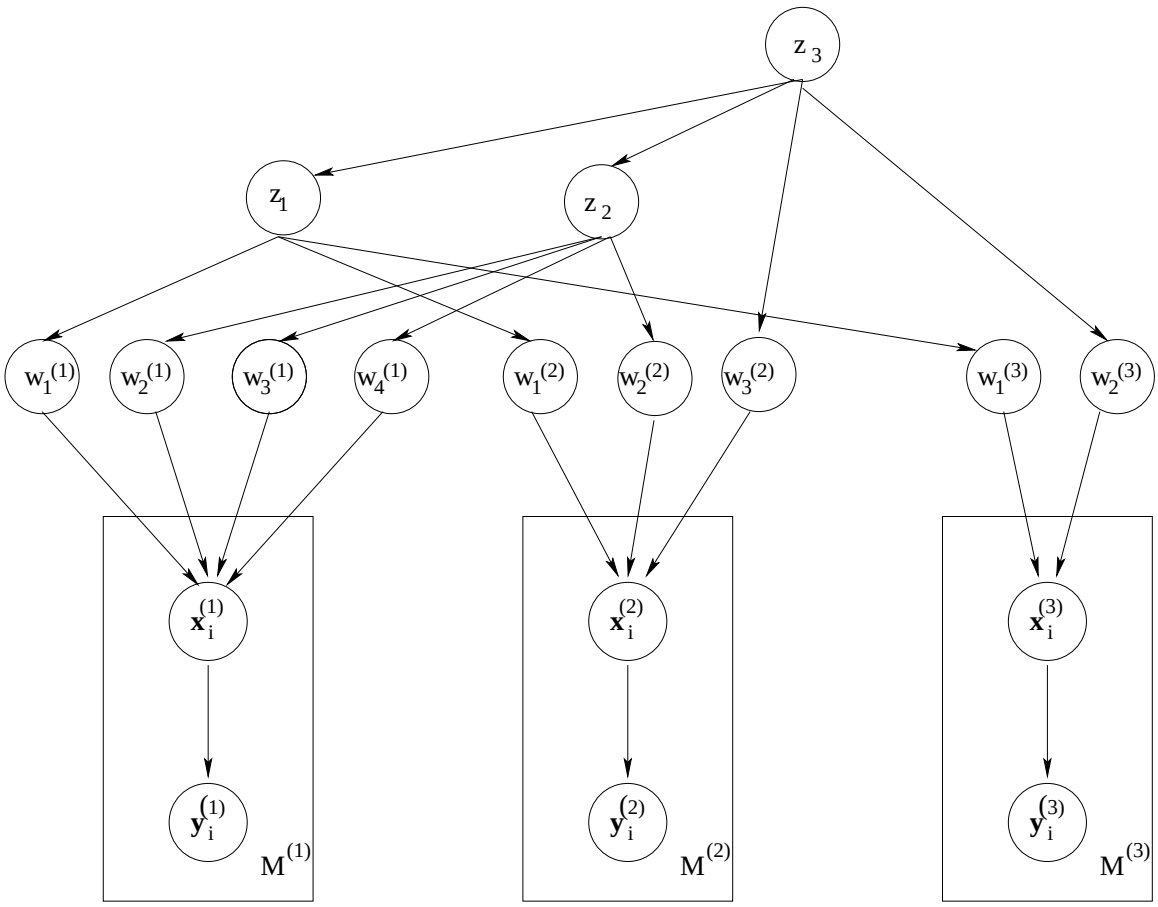


Figure 3.1: Graphical representation of the hierarchical transfer model.

being *(O)utside* or *(I)nside* person names, and we are given the labeled sample training sentence:

$$O \quad O \quad O \quad O \qquad O \qquad\qquad I$$

$$\texttt{Give the book to Professor Caldwell} \tag{3.1}$$

one such useful feature might be: *Is the token one slot to the left of the current token* ***Professor****?* We can represent this symbolically as *L.1.Professor* where we describe the whole space of useful features of this form as: {***direction*** *= (L)eft, (C)urrent, (R)ight*}.{***distance*** *= 1, 2, 3, ...*}.{***value*** *= Professor, book, ...*}. Some example features describable this way[1]

[1]Defining features in this form allows the natural language toolkit we use for these experiments, Mi-

CurrentToken.charPattern.Xx = TRUE
LeftToken.1.isTitle = TRUE
LeftToken.1.lowerCase.isWord.professor = TRUE

Table 3.1: Examples of features for token `Caldwell` in example Sentence 3.1.

are shown in Table 3.1 and further described in §4.1.1. We can conceptualize the structure of this feature space as a tree, where each slot in the symbolic name of a feature is a branch and each period between slots represents another level, going from root to leaf as read left to right. Thus a subsection of the entire feature tree for the token `Caldwell` could be drawn as in Figure 3.2 (zoomed in on the section of the tree where the *L.1.Professor* feature resides).

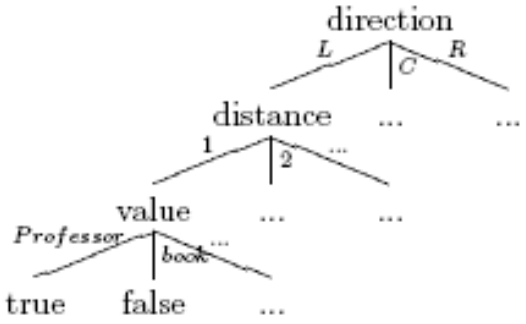


Figure 3.2: Graphical representation of a hierarchical feature tree for token `Caldwell` in example Sentence 3.1.

Representing feature spaces with this kind of tree, besides often coinciding with the explicit language used by common natural language toolkits [Cohen, 2004], has the added benefit of allowing a model to easily back-off, or smooth, to decreasing levels of specificity. For example, the leaf level of the feature tree for our sample Sentence 3.1 tells us that the word `Professor` is important, with respect to labeling person names, when located one slot to the

northird, to recursively instantiate tens of thousands of features based on a very simple set of user-defined patterns, such as *IsNumeral* or *IsTitle*. See Appendix A for more details.

left of the current word being classified. This may be useful in the context of an academic corpus, but might be less useful in a medical domain where the word `Professor` occurs less often. Instead, we might want to learn the related feature *L.1.Dr*. In fact, it might be useful to generalize across multiple domains the fact that the word immediately preceding the current word is often important with respect to the named entity status of the current word. This is easily accomplished by backing up one level from a leaf in the tree structure to its parent, to represent a class of features such as *L.1.**.

It has been shown empirically that, while the significance of *particular* features, such as *ThisToken.equals.mr* or *ThisToken.equals.professor*, might vary between domains and tasks, certain generalized *classes* of features, such as *ThisToken.IsTitle*, retain their importance across domains [Minkov et al., 2005].

3.1.2 New model: hierarchical prior model

One way of implementing this sort of "back-off" is to use the feature hierarchy as a prior for transferring beliefs about the significance of entire *classes* of features across domains and tasks. Some examples illustrating this idea are shown in Table 3.2. In these examples, the asterisk (*) stands for wildcard and will match anything. For example, the feature `LeftToken.IsWord.IsTitle.equals.*` would match any token which had a *title* directly to its left, while `LeftToken.IsWord.IsTitle.equals.mr` would only match tokens that has the specific token 'mr' on their left.

In this section, we will present a new model that learns simultaneously from multiple domains, by taking advantage of a feature hierarchy. We will assume that there are D domains on which we are learning simultaneously. Let there be M_d training data in each domain d. For our experiments with non-identically distributed, independent data, we use conditional random fields (cf. §2.2.4). However, this model can be used with any discriminative prob-

LeftToken.*
LeftToken.IsWord.*
LeftToken.IsWord.IsTitle.*
LeftToken.IsWord.IsTitle.equals.*
LeftToken.IsWord.IsTitle.equals.mr

Table 3.2: A few examples of the feature hierarchy

abilistic model, even those without sequential structure, such as the MaxEnt model. Let $\Lambda^{(d)} = (\lambda_1^{(d)}, \cdots, \lambda_{F_d}^{(d)})$ be the parameters of the discriminative model in the domain d where F_d represents the number of features in the domain d (while we focus on binary features in this work, this model is general enough to admit real valued features as well).

Further, we will also assume that the features of different domains share a common hierarchy represented by a tree $\mathcal{T}$, whose leaf nodes are the features themselves (cf. Figure 3.2). The model parameters $\Lambda^{(d)}$, then, form the parameters of the leaves of this hierarchy. Each non-leaf node $n \in$ non-leaf($\mathcal{T}$) of the tree (the w's of Figure 3.1) is also associated with a hyper-parameter z_n. Note that since the hierarchy is a tree, each node n has only one parent, represented by pa(n). Similarly, we represent the set of children nodes of a node n as ch(n).

The entire graphical model for an example consisting of three domains is shown in Figure 3.1. The conditional likelihood of the entire training data $(\mathbf{y}, \mathbf{x}) = \{(\mathbf{y}_1^{(d)}, \mathbf{x}_1^{(d)}), \cdots, (\mathbf{y}_{M_d}^{(d)}, \mathbf{x}_{M_d}^{(d)})\}_{d=1}^{D}$ is given by:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{z}) &= \left\{ \prod_{d=1}^{D} \prod_{k=1}^{M_d} P(\mathbf{y}_k^{(d)}|\mathbf{x}_k^{(d)}, \Lambda^{(d)}) \right\} \\ &\times \left\{ \prod_{d=1}^{D} \prod_{f=1}^{F_d} \mathcal{N}(\lambda_f^{(d)}|z_{\text{pa}(f^{(d)})}, 1) \right\} \\ &\times \left\{ \prod_{n \in \mathcal{T}_{nonleaf}} \mathcal{N}(z_n|z_{\text{pa}(n)}, 1) \right\} \end{aligned} \tag{3.2}$$

where the terms in the first line of eq. (3.2) represent the likelihood of data in each domain given their corresponding model parameters, the second line represents the likelihood of each model parameter in each domain given the hyper-parameter of its parent in the tree hierarchy of features and the last term goes over the entire tree $\mathcal{T}$ except the leaf nodes. Note that in the last term, the hyper-parameters are shared across the domains, so there is no product over d. Note also that the model described in eq. (3.2) is general: while for the remainder of the thesis we will often instantiate $P(\mathbf{y}_k^{(d)}|\mathbf{x}_k^{(d)}, \Lambda^{(d)})$ using conditional random fields (CRF), the method should apply equally well under the substitution of any *conditional* model.

We perform a MAP estimation for each model parameter as well as the hyper-parameters. Accordingly, the estimates' update rules are given as follows:

$$\begin{aligned} \lambda_f^{(d)} &= \sum_{i=1}^{M_d} \frac{\partial}{\partial \lambda_f^{(d)}} \left(\log P(\mathbf{y}_i^d|\mathbf{x}_i^{(d)}, \Lambda^{(d)}) \right) + z_{\mathrm{pa}(f^{(d)})} \\ z_n &= \frac{z_{\mathrm{pa}(n)} + \sum_{i \in \mathrm{ch}(n)} \lambda_i}{1 + |\mathrm{ch}(n)|} \end{aligned} \tag{3.3}$$

Essentially, in this model, the weights of the leaf nodes (model parameters) depend on the log-likelihood as well as the prior weight of its parent. Additionally, the weight of each hyper-parameter node in the tree is computed as the average of all its children nodes and its parent, resulting in a *smoothing* effect, both up and down the tree.

3.1.3 An approximate hierarchical prior model

The hierarchical prior model is a theoretically well founded model for transfer learning through feature hierarchy. In practice, however, it can be troublesome to compute. We therefore propose an approximate version of this model that weds ideas from the exact hierarchical prior model and the Chelba-Acero model.

As with the Chelba-Acero prior method in §2.2.4, this approximate hierarchical method also requires two distinct data sets, one for **training** the prior and another for **tuning** the

final weights. The tuning was performed by training a model to convergence on the tuning data set, and using the trained coefficients as the parameter values in the new model. Unlike Chelba-Acero, we smooth the weights of the priors using the feature-tree hierarchy presented in §3.1, like the hierarchical prior model.

For smoothing of each feature weight, we chose to back-off in the tree as little as possible until we had a large enough sample of prior data (measured as M, the number of subtrees below the current node) on which to form a reliable estimate of the mean and variance of each feature or class of features. For example, if the tuning data set is as in Sentence 3.1, but the prior contains no instances of the word `Professor`, then we would back-off and compute the prior mean and variance on the next higher level in the tree. Thus the prior for *L.1.Professor* would be $\mathcal{N}$(mean(L.1.*), variance(L.1.*)), where mean() and variance() of *L.1.** are the sample mean and variance of all the features in the prior dataset that match the pattern *L.1.** – or, put another way, all the siblings of *L.1.Professor* in the feature tree. If fewer than M such siblings exist, we continue backing-off, up the tree, until an ancestor with sufficient descendants is found.

This backing-off strategy has the result that the information contained in the data instances is kept closer to the leaves, based on the sample size for that leaf, which seems to be important. In fact, our preliminary experiments indicated that the approximate hierarchical model outperforms the exact model on real-life data. We conjecture that the main reason for this phenomenon is over-smoothing. In other words, by letting the information propagate from the leaf nodes in the hierarchy all the way to the root node, as in the exact method, the model loses its ability to discriminate between its features.

A detailed description of the approximate hierarchical algorithm is shown in Table 3.3. Notice the similarity to empirical Bayes techniques, where the height of our implicit underlying hierarchical Bayesian model varies depending on the sparsity of the data available to estimate

the parameters of our Gaussian prior.

It is important to note that this smoothed tree is an approximation of the exact model presented in §3.1.2 and thus an important parameter of this method in practice is the degree to which one chooses to smooth up or down the tree. One of the benefits of this model is that the semantics of the hierarchy (how to define a feature, a parent, how and when to back-off and up the tree, etc.) can be specified by the user, in reference to the specific datasets and tasks under consideration. For our experiments, the semantics of the tree are as presented in §3.1.1.

The Chelba-Acero method can be thought of as a hierarchical prior in which no smoothing is performed on the tree at all. Only the leaf nodes of the prior's feature tree are considered, and, if no match can be found between the tuning and prior's training datasets' features, a $\mathcal{N}(0, 1)$ prior is used instead. However, in the new approximate hierarchical model, even if a certain feature in the tuning dataset does not have an analog in the training dataset, we can always back-off until an appropriate match is found, even to the level of the root. As long as the hierarchy is constructed such that related features are near each other in the tree, this backing-off should result in a possibly weaker, but hopefully still relevant, estimate of the missing feature.

Henceforth, we will use only the approximate hierarchical model in our experiments and discussion.

Input: $\mathcal{D}^{source} = (X^{source}_{train}, Y^{source}_{train})$

$\mathcal{D}^{target} = (X^{target}_{train}, Y^{target}_{train})$;

Feature sets $\mathcal{F}^{source}$, $\mathcal{F}^{target}$;

Feature Hierarchies $\mathcal{H}^{source}$, $\mathcal{H}^{target}$

Minimum membership size M

Train CRF using $\mathcal{D}^{source}$ to obtain

feature weights Λ^{source}

For each feature $f \in \mathcal{F}^{target}$

Initialize: node $n = f$

While ($n \notin \mathcal{H}^{source}$

or $|\text{Leaves}(\mathcal{H}^{source}(n))| \leq M$)

and $n \neq root(\mathcal{H}^{target})$

$n \leftarrow \text{Pa}(\mathcal{H}^{target}(n))$

Compute μ_f and σ_f using the sample

$\{\lambda^{source}_i \mid i \in \text{Leaves}(\mathcal{H}^{source}(n))\}$

Train Gaussian prior CRF using $\mathcal{D}^{target}$ as data

and $\{\mu_f\}$ and $\{\sigma_f\}$ as Gaussian prior parameters.

Output:Parameters of the new CRF Λ^{target}.

Table 3.3: Algorithm for approximate hierarchical prior: $\text{Pa}(\mathcal{H}^{source}(n))$ is the parent of node n in feature hierarchy $\mathcal{H}^{source}$; $|\text{Leaves}(\mathcal{H}^{source}(n))|$ indicates the number of leaf nodes (basic features) under a node n in the hierarchy $\mathcal{H}^{source}$.

Table 3.4: Summary of data used in experiments

Corpus	Genre	Task	Tokens	Features	Frequency of positive class
UTexas	Bio	Protein	217,000	105,000	6.6%
Yapex	Bio	Protein	61,000	37,000	15.0%
MUC6	News	Person	45,000	40,000	2.29%
MUC7	News	Person	102,000	68,000	2.20%
CSPACE	E-mail	Person	28,000	19,000	4.20%

3.2 Investigation of hierarchical feature models

3.2.1 Data, domains and tasks

For our investigations into hierarchical feature models, we chose five different corpora (summarized in Table 3.4). Although each corpus can be considered its own *domain* (due to variations in annotation standards, specific task, date of collection, etc), they can also be roughly grouped into three different *genres*. These are: *abstracts from biological journals* [UT [Bunescu et al., 2004], Yapex [Franzén et al., 2002]]; *news articles* [MUC6 [Fisher et al., 1995], MUC7 [Borthwick et al., 1998]]; and *personal e-mails* [CSPACE [Kraut et al., 2004]]. Each corpus, depending on its *genre*, is labeled with one of two name-finding *tasks*:

- protein names in biological abstracts
- person names in news articles and e-mails

We chose this array of corpora so that we could evaluate our hierarchical prior's ability to generalize across and incorporate information from a variety of domains, genres and tasks.

In each case, each item (abstract, article or e-mail) was tokenized and each token was hand-labeled as either being part of a name (protein or person) or not, respectively. We used a standard natural language toolkit [Cohen, 2004] to compute tens of thousands of binary

features on each of these tokens, encoding such information as capitalization patterns and contextual information from surrounding words. This toolkit produces features of the type described in §3.1.1 and thus was amenable to our hierarchical prior model. In particular, we chose to use the simplest default out-of-the-box feature generator and purposefully did not use specifically engineered features, dictionaries, or other techniques commonly employed to boost performance on such tasks. The goal of our experiments was to see to what degree named entity recognition problems naturally conformed to hierarchical methods, and not just to achieve the highest performance possible.

3.2.2 Experiments & results

We evaluated the performance of various transfer learning methods on the data and tasks described in §3.2.1. Specifically, we compared our approximate hierarchical prior model (HIER), implemented as a CRF, against three baselines:

- GAUSS: CRF model tuned on a single domain's data, using a standard $\mathcal{N}(0,1)$[1] prior
- CAT: CRF model tuned on a concatenation of multiple domains' data, using a $\mathcal{N}(0,1)$[2] prior
- CHELBA-ACERO: CRF model tuned on one domain's data, using a prior trained on a different, related domain's data (cf. §2.2.4)

We use token-level $F1$ as our main evaluation measure, combining precision and recall into one metric. These results can be viewed in light of the similar experiments performed in §2.2.6. Specifically the Chelba-Acero model, which demonstrated a substantial win over the other methods in the inductive transfer setting, serves as a plausible baseline to the approximate hierarchical prior model evaluated here.

[2]We found anecdotal evidence suggesting these baselines were robust across a range of choices of default prior variance.

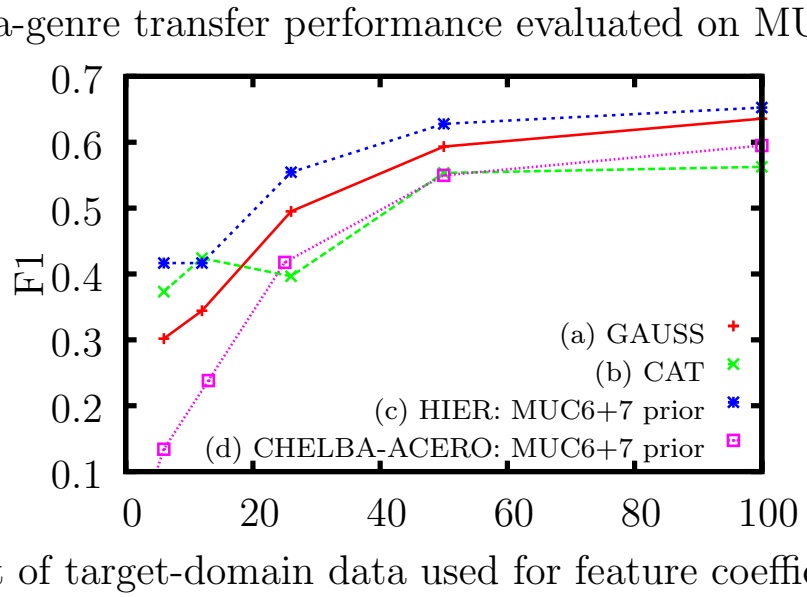


Figure 3.3: Adding a relevant HIER prior helps compared to the GAUSS baseline (**(c)** > **(a)**), while simply CAT'ing or using CHELBA-ACERO can hurt (**(d)** ≈ **(b)** < **(a)**, except with very little data), and never beats HIER (**(c)** > **(b)** ≈ **(d)**). All models were tuned on MUC6 except CAT (**b**), tuned on MUC6+MUC7.

3.2.3 Intra-genre, same-task transfer learning

Figure 3.3 shows the results of an experiment in learning to recognize person names in MUC6 news articles. In this experiment we examined the effect of adding extra data from a different, but related domain from the same genre, namely, MUC7. Line *a* shows the F1 performance of a CRF model tuned only on the target MUC6 domain (GAUSS) across a range of tuning data sizes. Line *b* shows the same experiment, but this time the CRF model has been tuned on a dataset comprised of a simple concatenation of the training MUC6 data from (*a*), along with a different training set from MUC7 (CAT). We can see that adding extra data in this way, though the data is closely related both in domain and task, has actually hurt the performance of our recognizer for *training sizes* of moderate to large size (the **x-axis** in the plot). This is most likely because, although the MUC6 and

MUC7 datasets are closely related, they are still drawn from different distributions and thus cannot be intermingled indiscriminately. Line c shows the same combination of MUC6 and MUC7, only this time the datasets have been combined using the HIER prior. In this case, the performance actually does improve, both with respect to the single-dataset trained baseline (a) and the naively trained double-dataset (b). Finally, line d shows the results of the CHELBA-ACERO prior. Curiously, though the domains are closely related, it does more poorly than even the non-transfer GAUSS. One possible explanation is that, although much of the vocabulary is shared across domains, the interpretation of the features of these words may differ. Since CHELBA-ACERO doesn't model the hierarchy among features like HIER, it is unable to smooth away these discrepancies. In contrast, we see that our HIER prior is able to successfully combine the relevant parts of data across domains while filtering the irrelevant, and possibly detrimental, ones.

This experiment was repeated for the three other sets of intra-genre tasks ($MUC6 \rightarrow MUC7$, $Yapex \rightarrow UT$ and $UT \rightarrow Yapex$), with the results shown in Figures 3.4, 3.5 and 3.6, respectively, and summarized in §3.2.5.

3.2.4 Inter-genre, multi-task transfer learning

In Figure 3.7 we see that the properties of the hierarchical prior hold even when transferring across tasks. Here again we are trying to learn to recognize person names in MUC6 e-mails, but this time, instead of adding only other datasets similarly labeled with person names, we are additionally adding biological corpora (UT & YAPEX), labeled not with person names but with protein names instead, along with the CSPACE e-mail and MUC7 news article corpora. The robustness of our prior prevents a model trained on all five domains (g) from degrading away from the intra-genre, same-task baseline (e), unlike the model trained on concatenated data (f). CHELBA-ACERO (h) performs similarly well in this case, perhaps

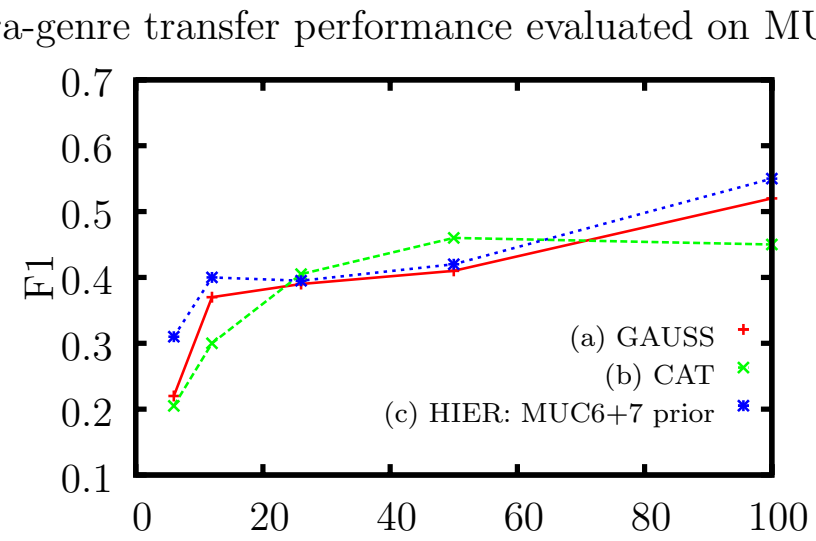


Figure 3.4: All models were trained on MUC6 and tuned on MUC7 except CAT (**b**), tuned on MUC6+MUC7.

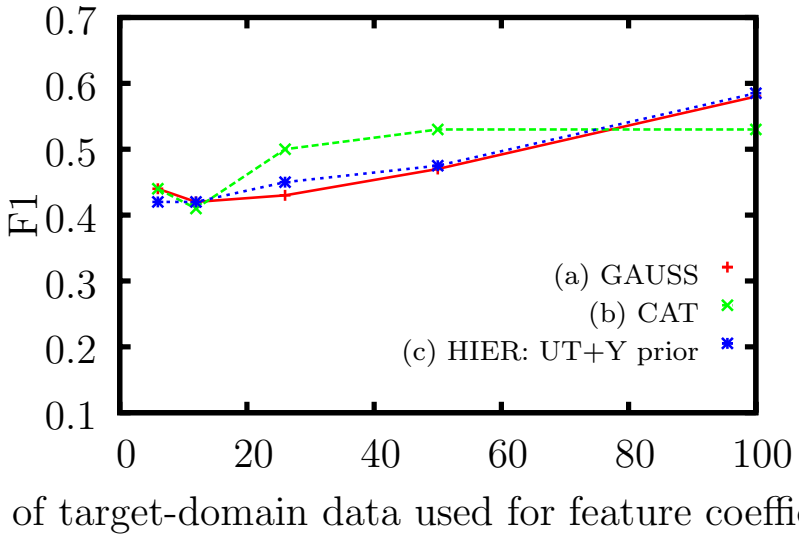


Figure 3.5: All models were trained on Yapex (Y) and tuned on UTexas (UT) except CAT (**b**), tuned on UT+Y.

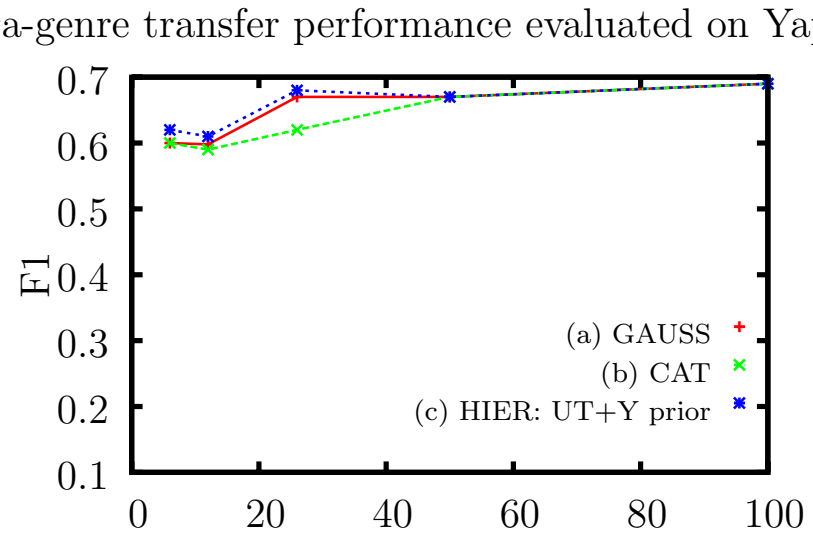


Figure 3.6: All models were trained on UTexas (UT) and tuned on Yapex (Y) except CAT (**b**), tuned on UT+Y.

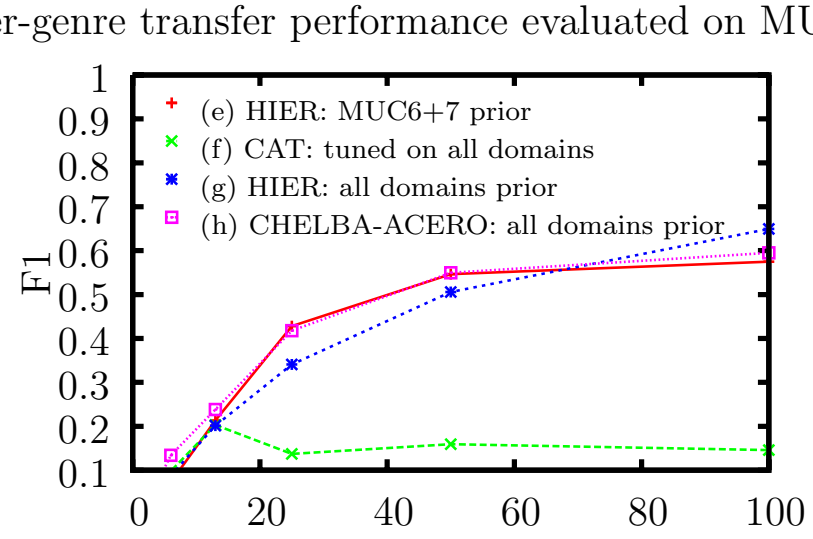


Figure 3.7: Transfer aware priors CHELBA-ACERO and HIER effectively filter irrelevant data. Adding more irrelevant data to the priors doesn't hurt ($(\mathbf{e}) \approx (\mathbf{g}) \approx (\mathbf{h})$), while simply CAT'ing it, in this case, is disastrous ($(\mathbf{f}) << (\mathbf{e})$. All models were tuned on MUC6 except CAT (**f**), tuned on all domains.

because the domains are so different that almost none of the features match between prior and tuning data, and thus CHELBA-ACERO backs-off to a standard $\mathcal{N}(0, 1)$ prior.

This robustness in the face of less similarly related data is very important since these types of transfer methods are most useful when one possesses only very little target domain data. In this situation, it is often difficult to accurately estimate performance and so one would like assurance than any transfer method being applied will not have negative effects.

3.2.5 Comparison of HIER prior to baselines

Each scatter plot in Figure 3.8 shows the relative performance of a baseline method against HIER (the full results, summarized in these scatter plots, are shown in Appendix B). Each point represents the results of two experiments: the y-coordinate is the F1 score of the baseline method (shown on the y-axis), while the x-coordinate represents the score of the HIER method in the same experiment. Thus, points lying below the y = x line represent experiments for which HIER received a higher F1 value than did the baseline.

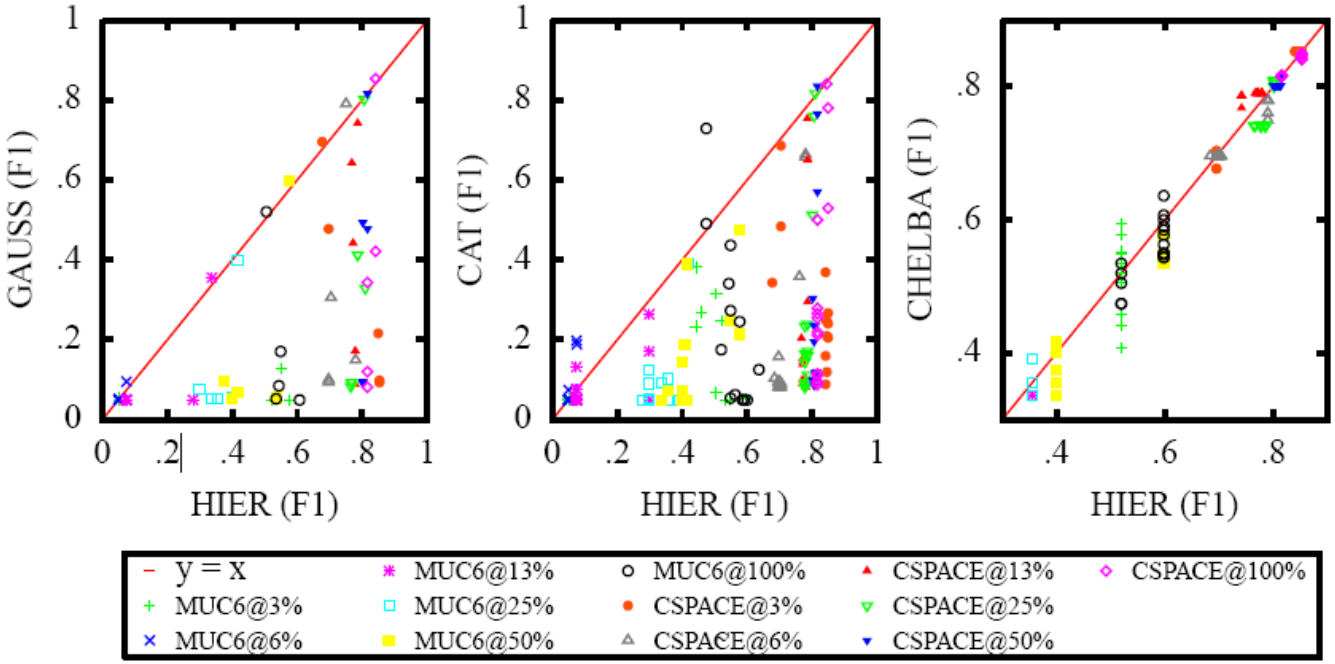


Figure 3.8: Comparative performance of baseline methods (GAUSS, CAT, CHELBA-ACERO) vs. HIER prior, as trained on nine prior datasets (both pure and concatenated) of various sample sizes, evaluated on MUC6 and CSPACE datasets. Points below the y = x line indicate HIER outperforming baselines.

While all three plots show HIER outperforming each of the three baselines, not surprisingly, the non-transfer GAUSS method suffers the worst, followed by the naive concatenation (CAT) baseline. Both methods fail to make any explicit distinction between the source and target domains and thus suffer when the domains differ even slightly from each other. Although the differences are more subtle, the right-most plot of Figure 3.8 suggests HIER is likewise able to outperform the non-hierarchical CHELBA-ACERO prior in certain transfer scenarios. CHELBA-ACERO is able to avoid suffering as much as the other baselines when faced with large difference between domains, but is still unable to capture as many dependencies between domains as HIER.

3.2.6 Prior work related to hierarchical feature models

While existing techniques have tried to quantify the generalizability of certain features across domains and used that to aid in transfer [Daumé III and Marcu, 2006; Jiang and Zhai, 2006], they do not provide an explicit, interpretable, set of priors which define and regulate what is meant by 'generalizable', as our feature hierarchy does. Other work has tried to exploit the common structure of related problems in the source and target domains [Ben-David et al., 2007; Schölkopf et al., 2005], but relies on labeled examples drawn from the target domain to do so, i.e., supervised transfer learning, while our work requires no labeled target data. While there are examples of unsupervised [Arnold et al., 2007], semi-supervised [Grandvalet and Bengio, 2005; Blitzer et al., 2006], and transductive approaches [Taskar et al., 2003], they likewise do not take advantage of the known, cross-domain, hierarchical relationship among features.

Recent work using so-called meta-level priors to transfer information across tasks [Lee et al., 2007], while related, does not take into explicit account the hierarchical structure of these meta-level features often found in NLP tasks. Daumé allows an extra degree of freedom

among the features of his domains, implicitly creating a two-level feature hierarchy with one branch for *general* features, and another for *domain specific* ones, but does not extend his hierarchy further [Daumé III, 2007]. More recent work extends and formalizes Daumé's two-level structure to a full Bayesian hierarchical model, allowing for more nuanced control of the relationship among domains in a more complex transfer task like our own [Finkel and Manning, 2009]. Finkel and Manning's work also presents a nice generalized framework from which to view our use of smoothing over hierarchies of linguistic features as a method for learning the parameters of a Gaussian regularization.

Work on hierarchical penalization [Szafranski et al., 2007] in two-level trees (concurrent with our ACL paper [Arnold et al., 2008]) tries to produce models that are parsimonious with respect to a relatively small number of *groups* of variables as structured by the tree, as opposed to transferring knowledge between and among the branches of the tree themselves, as in our transfer setting. Much of this hierarchical approach can also be related to wavelet-based methods [Donoho and Johnstone, 1995] that try to represent and compress the regularities in data using a known hierarchy. A key difference, however, is that wavelets tend to use a hierarchy of frequencies, useful for encoding images or sounds, and it is not clear how they would extend to categorical data such as tokens in a document.

3.2.7 Discussion

In this work we have introduced hierarchical feature tree priors for use in transfer learning on named entity extraction tasks. We have provided evidence that motivates these models on intuitive, theoretical and empirical grounds, and have gone on to demonstrate their effectiveness in relation to other, competitive transfer methods. Specifically, we have shown that hierarchical priors allow the user enough flexibility to customize their semantics to a specific problem, while providing enough structure to resist unintended negative effects when

used inappropriately. Thus hierarchical priors seem a natural, effective and robust choice for transferring learning across NER datasets and tasks.

From the broader perspective of this thesis as a whole, we have demonstrated that hierarchical feature trees provide a robust method for relating disparate parts of a data set to one another (in this case, features in feature space). The hierarchy provides a binding framework within which different aspects of the data can relate to and influence each other, and be aggregated by the learner to produce a model that is robust across these variations in the data.

Finally, while we have not investigated it here, we suspect these techniques for learning hierarchical priors could be applied to other structures besides trees, for example, polymorphic hierarchies or directed acyclic graphs (although there may be non-trivial issues such of semantics and convergence to address before such extensions could be achieved).

Chapter 4

Structural Frequency Features

In this chapter we define a novel feature based on the distribution of tokens across the structure of a document. We find that this feature has predictive properties that are preserved across domains, and thus provides a regularity that we can exploit to achieve more robust named entity recognition.

4.1 Definition of structural frequency features

Given a set of documents, each of which is **structured** into various sections, we can compute, for each token occurring in those documents, a statistic summarizing how often that token appears in one section of a document versus another. We call this segmentation of a data source into sections the document's *structure*, and the set of statistics gathered by conditioning on a token's distribution across the document's structure that token's **structural frequency features**. By modeling the distribution of instances across various related domains in a single unified feature space, structural frequency features are able to combine these disparate source of information in order to create a stronger learner [Arnold and Cohen,

2008]. This idea of using external, inter-dependent structure to improve learning robustness has been used previously by skip-chain conditional random fields to allow the incorporation of global, inter-connected constraints [Sutton and McCallum, 2004]. Previously, stacked learning introduced the idea of tying predictions together across examples to reduce bias and improve generalization performance [Wolpert, 1992], while more recent work has extended the stacked learning model to the specific problem of learning on sequentially related data common to many NER tasks [Cohen and Carvalho, 2005], as well as more arbitrary interactions, expressed graphically [Kou and Cohen, 2007].

4.1.1 Lexical features

Most modern information extraction systems rely on some kind of representation, usually a set of **features**, that distills the document into a form the algorithm can interpret and manipulate. The exact form of these features is a vital component of the overall system, balancing the complexity of a rich representation with the parsimony of an insightful view of the domain and problem being solved. For named entity recognition, **lexical features**, which try to capture patterns of words within the text of a document, are one of the most common, and intuitive, types of these representations. Generally, a lexical feature is a function of a word and its context. The specific definition of this function may vary widely across domains and implementations. In our setting, each lexical feature is a boolean function over a token in a document representing the value and morphology of that token and its neighbors. For example, given the sentence fragment from a caption of a biological paper: 'Figure 4: Tyrosine phosphorylation...', some lexical features for the token 'Tyrosine' would look like:

Notice that, although these features are defined with respect to a certain current token, 'Tyrosine', they also take into account the context of that word in the document. In

CurrentToken.isWord.Tyrosine
CurrentToken.charPattern.Xx
CurrentToken.endsWith.ine
Right1Token.endsWith.ation
Right1Token.isWord.phosphorylation
Left1Token.isWord.:
Left3Token.isWord.Figure

Table 4.1: Lexical features for token 'Tyrosine' in sample caption: 'Figure 4: *Tyrosine* phosphorylation...'.

this example, if we knew that this occurrence of 'Tyrosine' was labeled as a protein, the fact that the token immediately to the right of the current token was 'phosphorylation' (*Right1Token.isWord.phosphorylation*) might be useful in predicting whether other, heretofore unseen tokens besides 'Tyrosine', that also happen to be followed by a token such as 'phosphorylation', might also be proteins.

Since each word in one's vocabulary may constitute a feature (e.g., CurrentToken.isWord.A, CurrentToken.isWord.B, ...), it is not uncommon to have tens or even hundreds of thousands of such binary lexical features defined in one's feature space. The benefit of this is that such a large feature space can richly represent most any training set. The examples in Table 4.1 also include domain-specific features such as 'CurrentToken.endsWith.ine' (a common suffix for amino-acids). These custom features allow the researcher to bias his feature space towards specific features that he feels might be more informative with respect to his particular problem domain. While this specificity may be advantageous for an expert dealing with a limited domain, it can become a liability when that domain is uncertain, or even variable, as is the case in our robust learning setting.

For instance, while the occurrence of the word 'Figure' followed by a number and a colon may be very informative in terms of identifying words as proteins in the captions of papers, if our extractor is trained only on abstracts it may never see those types of features. Indeed, since lexical features are merely functions of the specific sections of text seen during training, they are unable to capture information residing in other sections of the document which may prove useful. Even in the semi-supervised case where the learning algorithm has access to unlabeled target domain data, lexical features are unable to take advantage of this information since there is no way to relate the unlabeled tokens to the labeled ones.

Lexical features thus provide a valuable, but brittle, representation of the training data. Our work augments these rich, though domain-specific, lexical features with other non-lexical features based on the internal structure of a document, contributing another view of the data that is more robust to changes in the domain. We show that combining these types of domain-specific and domain-robust features produces a classifier that performs well across domains.

4.1.2 Document structure

We begin by highlighting the common observation that most documents are written with some kind of internal structure. For instance, the biological papers we studied in this experiment (like most academic papers) can be divided into three sections:

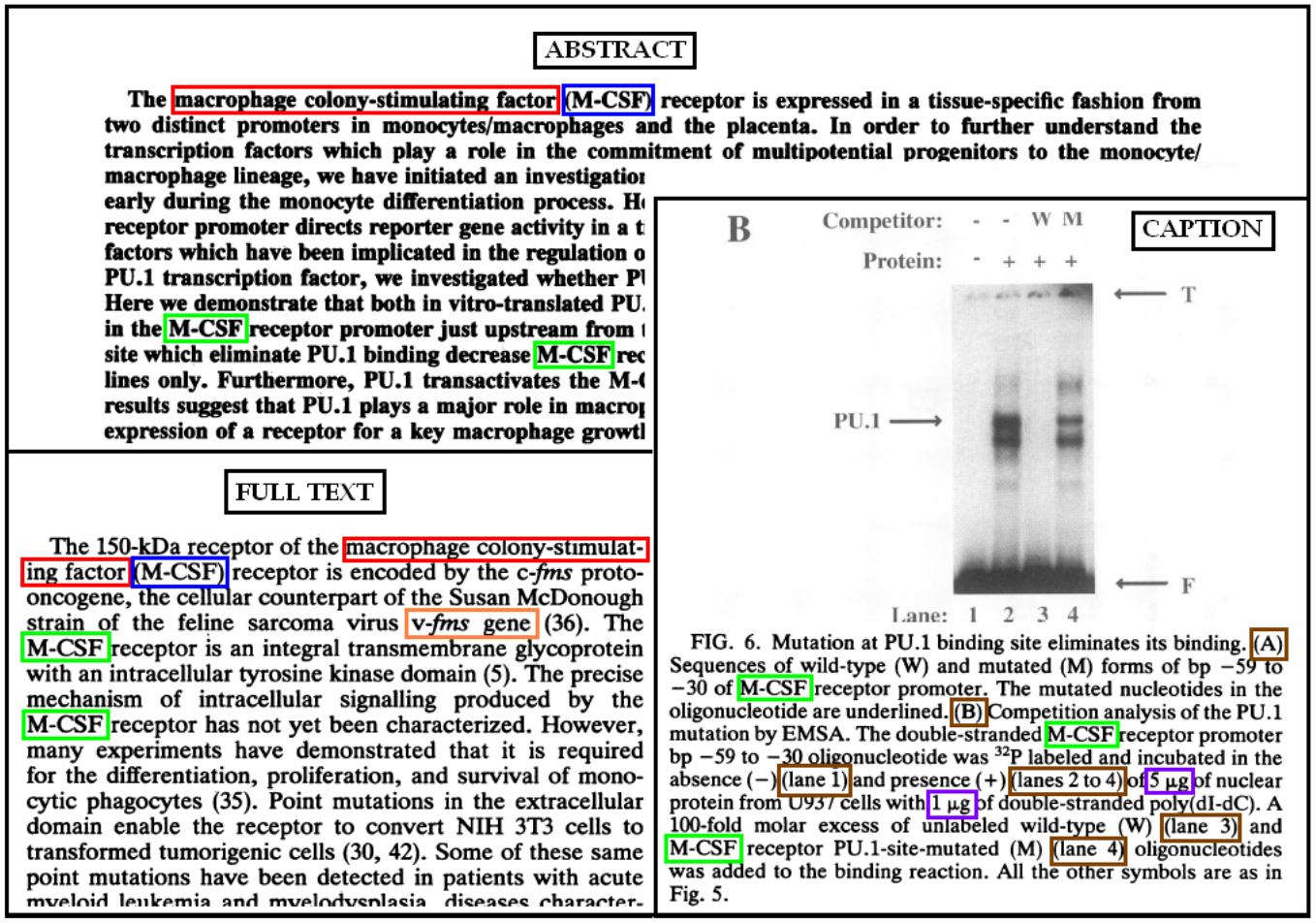
- **Abstract**: summarizing, at a high level, the main points of the paper such as the problem, contribution, and results.

- **Caption**: summarizing the figure it is attached to. These are especially important in biological papers where most important results are represented graphically. Unlike computer science papers, which usually have brief captions, in our corpus the average

caption was over 125 words long, thus supporting our belief that they might contain useful information for our NER task.

- **Full text**: the main text of a paper, that is, everything else besides the abstract and captions.

An example of such a structured document is provided in Figure 4.1. In this figure we see the various ways a protein can be referred to throughout the sections of a document.



ABSTRACT

The macrophage colony-stimulating factor (M-CSF) receptor is expressed in a tissue-specific fashion from two distinct promoters in monocytes/macrophages and the placenta. In order to further understand the transcription factors which play a role in the commitment of multipotential progenitors to the monocyte/ macrophage lineage, we have initiated an investigatio early during the monocyte differentiation process. H receptor promoter directs reporter gene activity in a t factors which have been implicated in the regulation o PU.1 transcription factor, we investigated whether P Here we demonstrate that both in vitro-translated PU. in the M-CSF receptor promoter just upstream from site which eliminate PU.1 binding decrease M-CSF rec lines only. Furthermore, PU.1 transactivates the M- results suggest that PU.1 plays a major role in macro expression of a receptor for a key macrophage growt

FULL TEXT

The 150-kDa receptor of the macrophage colony-stimulating factor (M-CSF) receptor is encoded by the c-*fms* proto-oncogene, the cellular counterpart of the Susan McDonough strain of the feline sarcoma virus v-*fms* gene (36). The M-CSF receptor is an integral transmembrane glycoprotein with an intracellular tyrosine kinase domain (5). The precise mechanism of intracellular signalling produced by the M-CSF receptor has not yet been characterized. However, many experiments have demonstrated that it is required for the differentiation, proliferation, and survival of monocytic phagocytes (35). Point mutations in the extracellular domain enable the receptor to convert NIH 3T3 cells to transformed tumorigenic cells (30, 42). Some of these same point mutations have been detected in patients with acute myeloid leukemia and myelodysplasia diseases character-

B

Competitor: - - W M

Protein: - + + +

T

PU.1

F

Lane: 1 2 3 4

CAPTION

FIG. 6. Mutation at PU.1 binding site eliminates its binding. (A) Sequences of wild-type (W) and mutated (M) forms of bp −59 to −30 of M-CSF receptor promoter. The mutated nucleotides in the oligonucleotide are underlined. (B) Competition analysis of the PU.1 mutation by EMSA. The double-stranded M-CSF receptor promoter bp −59 to −30 oligonucleotide was ^{32}P labeled and incubated in the absence (−) (lane 1) and presence (+) (lanes 2 to 4) of 5 μg of nuclear protein from U937 cells with 1 μg of double-stranded poly(dI-dC). A 100-fold molar excess of unlabeled wild-type (W) (lane 3) and M-CSF receptor PU.1-site-mutated (M) (lane 4) oligonucleotides was added to the binding reaction. All the other symbols are as in Fig. 5.

Figure 4.1: Sample biology paper. Each large black box represents a different subsection of the document's structure: abstract, caption and full text. Each small highlighted color box represents a different type of information: full protein name (red), abbreviated protein name (green), parenthetical abbreviated protein name (blue), non-protein parentheticals (brown), genes (orange), and measurement units (purple).

Notice how the distribution of these types of occurrences varies across the structure of the document. For instance, *full name references* (red) (like 'macrophage colony-stimulating factor') appear in the *abstract* and *full text* of the paper, but not its *caption*. In contrast, *non-protein parentheticals* (brown) (like '(A)', '(B)', '(lane 1)', '(lanes 2 to 4)', '(lane 3)', and '(lane 4)') do appear in the *caption* but not in the *full text* or *abstract*.

This is similar to the complex way the instances in Figure 1.1 are related to each other: not through a common distribution (as in the i.i.d. case), but rather through another mediating relationship (in this case, the structural features relating the occurrence of tokens across the **common** structure of a document). Here we see the importance of explicitly modeling the difference between the source and target domains: if one were to naïvely train a purely lexical feature based extractor on the abstracts and try to apply it to the captions, the extractor might be confused by the non-protein parentheticals, having never seen them in its training data. Likewise, it might waste significant probability mass on features representing the unabbreviated form of protein names which it might never see in its caption test data.

It is important to note that in order to support this interpretation of the data in which we can compare and aggregate token occurrences across different sections of the document, we have to make the so-called *one-sense-per-discourse* assumption [Gale et al., 1992]. This common assumption states that tokens in one section of a document have the same meaning as identical tokens in other sections of the same document. This can be visualized as another layer of edges in Figure 1.1, linking occurrences of words across sections of a document, and ultimately, bridging the gap between the source and target domains. This assumption is necessary since, without it, we would have no reason for believing that a potentially ambiguous token, such as 'CAT', used in a certain sense in one section of a document, would have the same sense in a different section of the document, and therefore, would have no way to aggregate that token's features and statistics across the entire document.

Since we have no labeled target domain data, however, it is not obvious how we might amend or supplement our source domain training data so as to avoid these problems. The key insight is the fact that these domains, while distinct, are nevertheless related by the overarching structure of the documents in which they reside. For instance, while unabbreviated protein names never appear in the caption, and non-protein parentheticals never appear in the abstract, both of these occur in the full text of the paper. Thus, our goal is to find some class of features that can relate these different types of occurrences together across the differing subsections of a document's structure. We will achieve this by leveraging the one-sense-per-discourse assumption and our knowledge about our documents' structure.

4.1.3 Structural frequency features

Let $D_1, D_2, \cdots, D_k$ be the k parts of text document D. Let $c(f, D_i)$ be the frequency count of feature f in D_i. A structural frequency feature is formally defined as: $c(f, D_i)/c(f, D_j)$. Like lexical features, structural frequency features are simply functions of tokens in context. Unlike purely lexical features, however, structural frequency features *are* able to leverage the occurrence of tokens across all sections of a document, including the unlabeled captions and full text. The idea is to leverage the fact that different types of tokens (e.g., unabbreviated protein names, non-protein parentheticals, etc.) occur with different frequencies in different sections of a document. In this sense, structural frequency features are related to the information theoretic concepts of conditional entropy and mutual information. In the example from Figure 4.1 in §4.1.2, we noticed that non-protein parentheticals occurred quite often in the caption, but not at all in the abstract. While this seems informative, in our setting, unfortunately, we do not have labels for the caption data. We are therefore unable to make a distinction between *protein* and *non-protein* parentheticals in the caption section of the document. We can, however, make such a distinction in the *abstract* section of the same

document, for which we do have labels. Thus, if we see a parenthesized token in a caption, and see the same token parenthesized in the abstract, we might be able to transfer that abstract token's label to the unlabeled caption occurrence. In this respect, these structural frequency features provide the links necessary to perform a kind of label propagation across the subsections of a document [Zhu and Ghahramani, 2002].

Given our previously stated one-sense-per-discourse assumption, we now have a means of transferring our labels across the different unlabeled sections of a document and may have a useful, non-transfer, semi-supervised learning model. Our ultimate goal, however, is semi-supervised domain adaptation, and these structural features, as described thus far, still lack a way of ensuring they will be robust across shifts in domain. The key to addressing that issue is to consider the occurrence of tokens not in isolation within each subsection of a document, but rather jointly across sections. For instance, in Figure 4.1 we see the pattern '(lane *)' occurs quite often in the caption, but never in the full text. In fact, there are many such non-proteins that only ever appear in the caption section of the document. In contrast, the token 'M-CSF' occurs with high frequency across all three sections of the document. Indeed, there are relatively few proteins that do *not* occur in the abstract of a paper.

It seems we can use the relative distribution of tokens across the different sections of a document, in and of itself and without any lexical or morphological information about the form of token itself, as a signal of that token's likelihood of being a protein. This makes sense, since authors are conveying different kinds of information, in different ways, across the various sections of a document and so are not equally likely to mention a protein, in the same particular way, across the entire document.

Specifically, for each unique word-type in a document, we counted the number of times it appeared in each of the different sections of that document (for example, the word-type 'M-CSF' occurs three times in the abstract, four times in the full text, and three times in the

	Times in:			Log prob. in:			Log cond. prob. in:	
Word	A	C	F	A	C	F	P(C|A)	P(F|A)
'M-CSF'	3	3	4	-1.84	-1.61	-3.10	-1.20	-1.12
'macrophage'	2	0	1	-2.01	-Inf	-3.70	-Inf	-1.72
'(M-CSF)'	1	0	1	-2.30	-Inf	-3.70	-Inf	-1.72
'PU.1'	5	2	0	-1.61	-1.78	-Inf	-1.37	-Inf
'kDa'	0	0	1	-Inf	-Inf	-3.70	Undefined	Undefined

Table 4.2: Sample structural frequency features for *specific* tokens in example paper from Figure 4.1, as distributed across the (A)bstract, (C)aptions and (F)ull text. Log probabilities are computed assuming the following number of *total* tokens are found in each section of the paper: $A = 206$, $C = 121$, $F = 4,971$, $C|A = 47$, $F|A = 53$.

caption of the example in Figure 4.1). We then normalized these counts by the total number of tokens in a given section to come up with an empirical probability of a word-type occurring in a particular section. We also computed the conditional forms of these features, that is, we counted the number of times a token appeared in section x, given that it also appeared in section y, again normalizing to form an empirical probability distribution. Continuing our example, the token 'macrophage' never occurs in the caption and thus, although the token does occur in the abstract, probability(word occurring in caption|word occurs in abstract) is still zero (see Table 4.2 for more examples). These conditional structural frequency features allow us to characterize the particular distribution patterns that different types of words have across the sections of a document. In particular, we might be interested in modeling things like p(word is a protein|word appears in caption but not in abstract). Figures 4.2 and 4.3 show the distribution of two such features across our training data.

Figure 4.2 shows a histogram of the number of times words labeled in the *abstract* as pro-

teins (left) and non-proteins (right) occurred with a given log normalized probability in the document's full text, given that it also appeared (at least once) in the same document's abstract section. Since these probabilities are plotted on the log scale, any zero values (i.e., words that appear in abstracts but never in the full text), will be assigned to the bin at

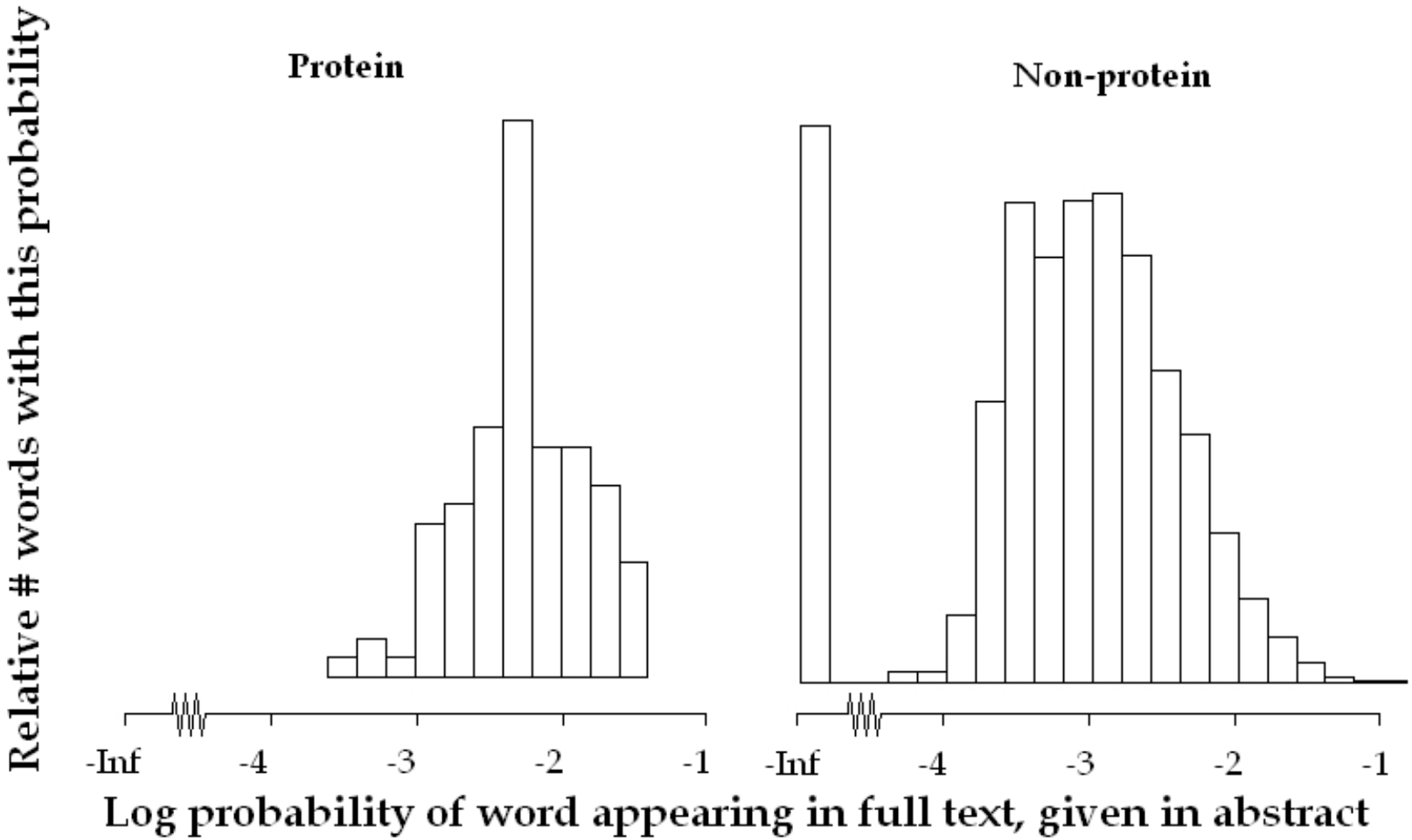


Figure 4.2: Histogram of the number of occurrences of protein (left) and non-protein (right) words with the given log normalized probability of appearing in *full text*, given that they also appear in an article's *abstract*.

negative infinity. The lack of instances at negative infinity in the left plot is evidence that, if a protein is in an abstract, it is also always in the full text at least once. But this is not so for non-proteins – the large spike on the left side of the right plot shows a large number of non-proteins that appear in abstracts but never in the full text. Also notice the general right-shift of the entire distribution in the left plot, indicating an overall higher proportion of proteins occurring in full-text, given that they appear in an abstract, than non-proteins.

Figure 4.3 shows a similar distribution, only this time the conditional structural frequency feature is measuring the likelihood of a word occurring in the *captions* of a paper, given that it appeared in the abstract. Notice, again, the left spike in the non-protein histogram on the right, indicating that a large number of non-proteins never appear in article's captions, despite appearing in its abstract. In contrast, the higher peaks to the right of the protein plot on the left show a much higher proportion of proteins appearing in captions, given they also appear in the abstract.

These plots clearly demonstrate a significant difference in the distribution of protein and non-protein tokens across the various subsections (abstract, captions, and full text) of a document's structure and suggest these structural frequency features may be informative with respect to identifying and extracting proteins. Thus, at training time, we compute these structural frequency features for each token in our labeled training abstracts. Since counting token occurrences across document sections, however, does not require labels itself, we can freely use all the unlabeled text from the papers we have to calculate the features. Likewise, by leveraging the one-sense-per-discourse assumption, we can attach the word-type's label (found in the abstract) to each of these features defined across the various sections of the document. In the end, we are left with a semi-supervised intra-document representation of the labeled abstract data that is, due to its cross structural nature, robust to shifts across the various document section domains.

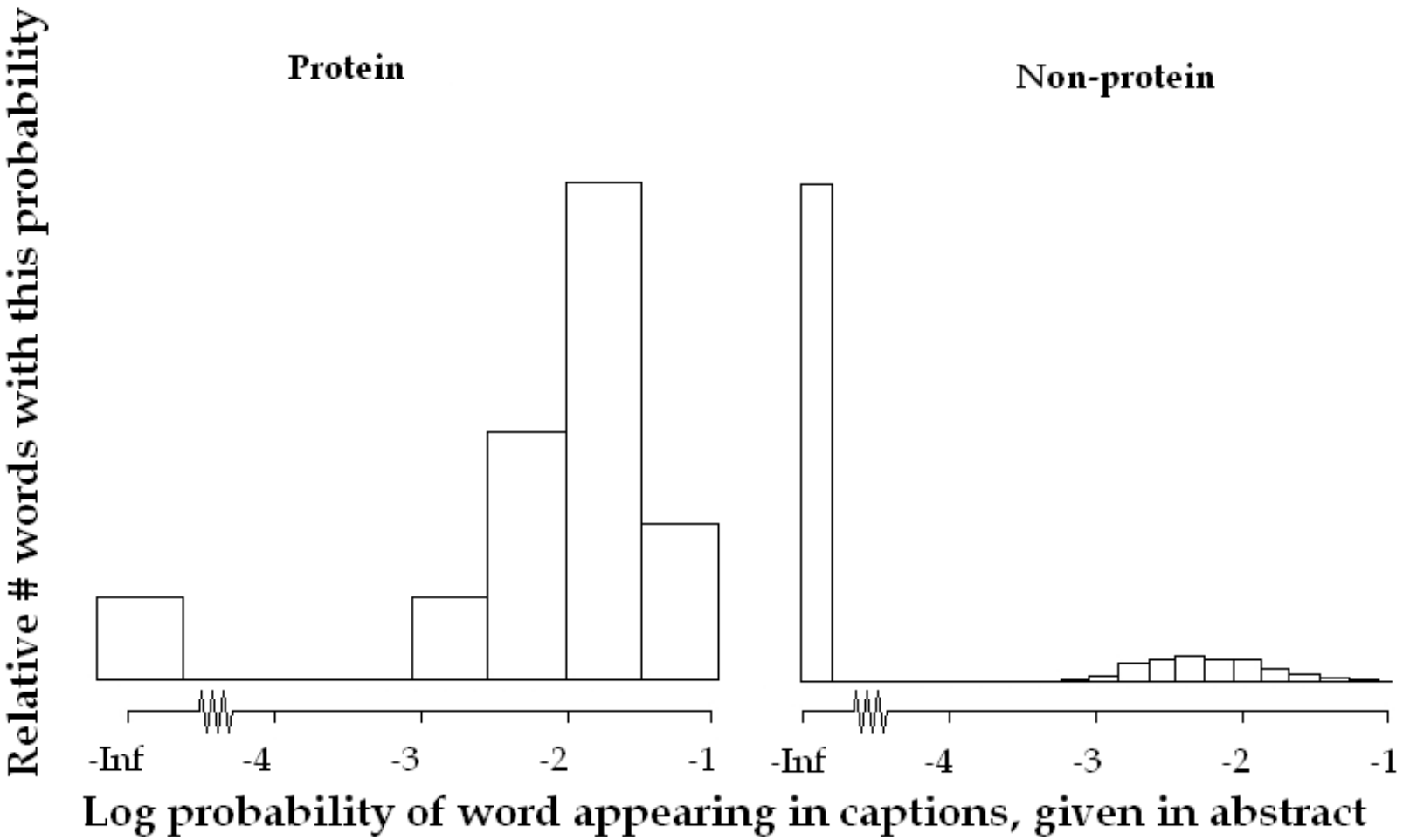


Figure 4.3: Histogram of the number of occurrences of protein (left) and non-protein (right) words with the given log normalized probability of appearing in *captions*, given that they also appear in an article's *abstract*.

4.2 Investigation of structural frequency features

4.2.1 Data

Our training data for these experiments was drawn from two sources:

- GENIA: a corpus of Medline abstracts with each token annotated as to whether it is a protein names or not [Ohta et al., 2002]
- PubMed Central (PMC): a free, on-line archive of biological publications [National Institues of Health, 2009]

Since our methods rely on having access to a document's labeled abstract along with the unlabeled captions and full text, and GENIA[1] only provided labeled abstracts, we had to search PMC for the corresponding full text, where available. Of GENIA's 1,999 labeled abstracts, we were able to find the corresponding full article text (in PDF format) for 303 of them on PMC. These PDF's were (noisily) converted to text[2] and segmented into abstract, captions, and full text using automated tools. Figure 4.1 shows an example of one such segmented PDF.

Of these 303 papers, consisting of abstracts labeled with protein names along with corresponding unlabeled captions and full text, 218 (consisting of over 1.5 million tokens) were used for training, and 85 (almost 640,000 tokens) were used for testing. From these documents we computed the previously described standard *lexical features*, along with 12 different *structural frequency feature* statistics (FREQ) for each unique token in the corpus, summarizing that token's conditional distribution in both *protein* and *non-protein* classes across

[1]Of the biological journal corpora used in §3.2.1, only Genia could be used for these experiments since the UTexas abstracts were not labeled with their corresponding PubMed id numbers, and the Yapex abstracts, while labeled with paper ids, did not have their full text available in PMC

[2]e-PDF PDF to Text Converter v2.1: http://www.e-pdfconverter.com

the abstract, captions, and full text of the document, corresponding to the $D_{abstract}$, $D_{caption}$ and $D_{fulltext}$ different sections of a document, as specified in our formal structural frequency feature definition. These features were then provided as training data to a CRF-based extractor, with evaluations performed on held-out data via cross validation.

4.2.2 Experiment & results

Non-transfer: abstract to abstract

In this non-transfer experiment, our standard CRF-based model labeled tokens of held-out *abstracts* as protein or not, and these predictions were automatically evaluated with respect to token-level precision, recall and F1 measure using the held-out GENIA labels for those abstracts. Figure 4.4 compares the performance of extractors trained only on lexical features (**LEX** of §4.1.1), only on structural frequency features (**FREQ** of §4.1.3), and on a combination of both types of features (**LEX+FREQ**), while Table 4.3 summarizes the precision, recall and F1 values of these models' as evaluated over the test data.

We can observe that, while the lexically trained model always outperforms the strictly structural frequency informed model (LEX dominates FREQ), the FREQ model nevertheless produces a competitive precision-recall curve despite having no access to any lexical information. This supports the intuition developed from observing the difference between protein and non-protein distributions in Figures 4.2 and 4.3.

Similarly, the fact that the combined model LEX+FREQ dominates each constituent model (LEX and FREQ individually) demonstrates that each type of feature (lexical and structural) is contributing a share of unique information, not represented by the other. This supports the connection with co-training, proposed in §5.1, by indicating that the feature sets are somewhat independent with respect to identifying protein names. The fact that their effect

in the combined model is not completely additive suggests they are not wholly independent either.

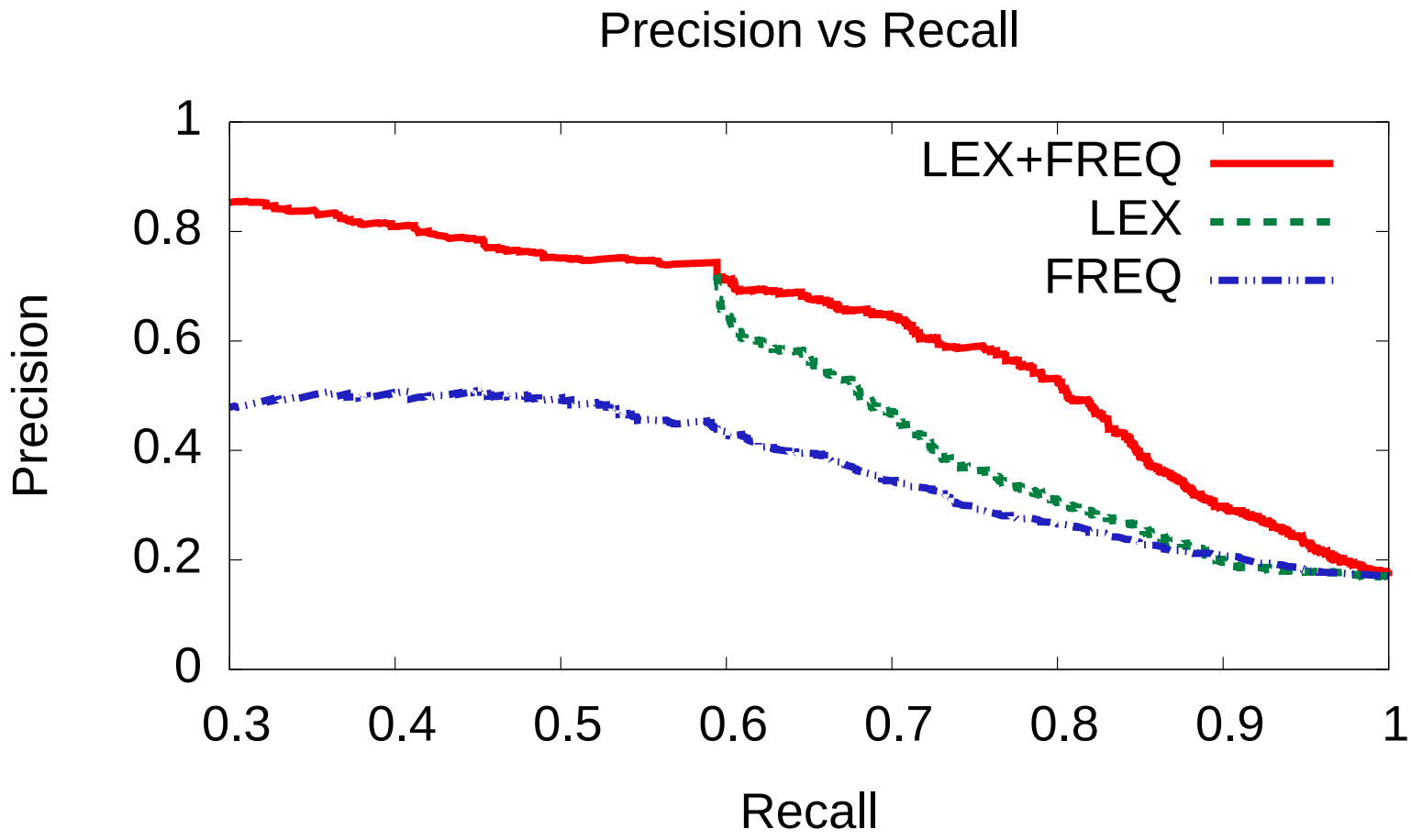


Figure 4.4: Precision versus recall of extractors trained on only lexical features (LEX), only structural frequency features (FREQ), and both sets of features (LEX+FREQ).

Model name	Prec	Rec	F1
LEX+FREQ	.74	.64	**.69**
LEX	.75	.55	.64
FREQ	.48	.58	.53

Table 4.3: Summary of results for extractors trained on full papers and evaluated on *abstracts*. Values in **bold** are significantly greater than those in plain font (one-sided paired t-test, $p < .01$).

Transfer: abstract to caption

Cross-domain experiments involving structural frequency features (FREQ) are fully described in §5.2.4 and 5.2.5, where they are presented in the context of complete ablation studies, along with the soon-to-be-introduced *snippets* of §5.1.

Conclusions

A concluding discussion of structural frequency features is likewise defered until §5.3, so as to incorporate the closely related concept of *snippets*.

Chapter 5

Snippets

Although structural frequency features provide domain-robust signals to our extractor, they do not directly ameliorate the domain-brittleness of the lexical features discussed in §4.1.1.

5.1 Definition of snippets

To address this issue, we introduce a kind of pseudo-data we call snippets. **Snippets** are tokens or short phrases taken from one of the unlabeled sections of a document and added to the training data, having been automatically positively or negatively labeled by some high confidence method [Arnold and Cohen, 2008]. Together, they help make the target distribution 'look' more like the source distribution with respect to the characteristics they share, while reshaping the target distribution away from the source distribution in regards to the ways in which they differ. The net effect is to produce an augmented view of the training data that will produce a more robust learner. We achieve this robustness by leveraging a key assumption: that tokens that commonly appear near words of a certain class (*protein* or *non-protein*) in source text will also tend to be neighbors of *similarly* classed words in the

target text.

In this way, snippets are related to previous work which dealt with creating pseudo-labeled data based on limited domain knowledge or weak constraints. In particular, Wang *et al.* exploit the weakly labeled tags associated with biological article abstracts to increase the amount of annotated data available to a learner [Wang et al., 2008], while Daumé uses auxiliary data from related tasks, along with prior knowledge about the relationships between and consistency constraints among these tasks, in order to synthesize pseudo-labeled data, or *hints*, which are shown to aid the learning process [Daumé III, 2008], in a process akin to bootstrap learning.

5.1.1 Positive snippets

Positive snippets (i.e., snippets automatically labeled as positive examples) are an attempt to leverage the overlap between and across domains, by taking high confidence examples from one domain and transferring them to the other. In this sense, it is related to co-training [Blum and Mitchell, 1998]. Specifically, positive snippets leverage the one-sense-per-discourse assumption (which we again rely upon due to our lack of labeled target data).

The procedure for generating positive snippets is relatively straight-forward:

1. All positively labeled tokens are extracted from the labeled *source* sections of the document (in this experiment, these are *proteins* in the *abstracts*), or encoded via *a priori* domain knowledge (such as a dictionary or gazetteer).

2. The unlabeled target sections of the document are searched for these positive tokens (having been extracted from the labeled source sections in step 1).

3. Any matching instances are copied from the unlabeled section, along with a bit of neighboring context (we use a default of three neighboring tokens on each side), directly

into the training data (by concatenating at the end).

4. These copied sections of text (called *snippets*), having been copied into the source training data, are then pseudo-labeled using the one-sense-per-discourse assumption: the snippet tokens matching other positively labeled tokens from the labeled source-data sections are labeled *positive*, while their neighboring context tokens (where they do **not** match a protein name observed in the source data) are left unlabeled, and therefore, implicitly negative.

5. This modified training data, now containing pseudo-labeled snippets from the target data, is then passed to the learner as usual.

The idea behind this process is that the surrounding context will help inform the extractor of the differences in the distribution of lexical features between the source and target domains. Since our goal is to train an extractor that will be robust to shifts from source to target domain, we would like to introduce some examples of the target domain into the source domain training data to make it look more like the target domain. Since we don't have labels for the target domain, however, we have to rely on this high-confidence (albeit possibly low recall) token matching heuristic and the assumption that, in the absence of other information such as dictionaries and gazetteers, unlabeled context surrounding pseudo-labeled snippets contains only negative tokens.

Although we focus in this work on specific methods for pseudo-labeling our examples, and learning algorithms for generalizing from these pseudo-labels, we believe the idea behind *snippets* should be generalizable to a wider class of domains and techniques, besides named entity recognition in text using discriminative classifiers. The exact form and method in which snippets are constructed will depend on the specifics of the domain being studied, but in general, the practitioner will want to optimize the performance of the techniques being used to pseudo-label the data (whether classifiers, stop-lists, dictionaries, etc) to the

characteristics of the problem, for example, increasing precision at the expense of recall when the cost of a false positive is disproportionately high.

5.1.2 Negative snippets

Similar to the positive variety, **negative snippets** (i.e., snippets automatically labeled as negative examples) provide examples of tokens which may appear to be proteins when viewed with respect to the source domain, but are in fact not proteins in the target domain. These must rely on some form of prior knowledge about the target domain for their high-confidence automatic labeling, perhaps some kind of extractor previously trained for the target domain or a gazetteer. For example, a researcher may have previously trained an extractor to identify tokens in captions that refer to specific panel locations in the accompanying image (e.g., the token '(B)' in Figure 4.1's caption). We call these types of references *image pointers* [Cohen et al., 2003]. Although this kind of token pattern may look like a parenthetical protein mention if seen in an abstract, since we have an existing extractor able to identify it as an image pointer in captions (and thus, by assumed mutual exclusion, not a protein), we are able to add all occurrences in a paper's captions of similarly identified image pointers (labeled as negative) to that paper's labeled training data. A similar process can be followed for all kinds of high-confidence negative labels, such as bibliographic citations, lists of measurement units, and various other stoplists.

Given a list of high-confidence *negative* tokens collected in this way (or, equivalently, extractors trained to detect them), *negative snippets* can be constructed in a way analogous to *positive snippets*. Specifically, unlabeled tokens from the target data are matched against the list of collected negative tokens (extractors) and copied over into the source training data along with their context, as before. In contrast to *positive snippets*, however, the entire snippet (both the matched negative token and its surrounding context) are implicitly

pseudo-labeled as negative examples. This is done because we do not have any labels for the target data (except any *positive snippet* lists) and so cannot tell if any of the negative token's context belong to the POSITIVE class. In the absence of this information, our default choice is to leave the snippets implicitly labeled as negative, since this is the most likely guess in the absence of other information. While this may lead to false negatives in the pseudo-labeled training data, it will nevertheless allow us to use our unlabeled target data not just to add new inter-domain information (as with structural frequency features), but also, perhaps as importantly, to adjust and augment the distribution of existing source domain derived lexical features to make them more in accord with the target domain, ultimately producing extractors that are more robust to changes between training and test domains.

5.2 Investigation of snippets & structural frequency features

We now examine the utility of our two new types of features:

- **Structural frequency features**: Informative with respect to protein extraction, but make repeated occurrences of the same token in different sections look similar.
- **Snippets**: Pseudo-examples that push a learned classifier towards being consistent with the one-sense-per-discourse assumption.

5.2.1 Data

For these experiments we used the same data as for the structural frequency feature experiments in Section 4.2.

5.2.2 Experiment

We used ablation studies to assess the amount of information our novel features each contribute to the task of protein name extraction, both in the non-transfer (abstract to abstract) and domain adaptation (abstract to caption) setting. In each case, we trained an extractor on a version of the training data constructed with the appropriate set of features:

- Structural frequency features (FREQ): As described in §4.1.3.

- Positive snippets (POS): As described in §5.1.1, high-confidence positively pseudo-labeled examples of tokens (i.e., *proteins*), extracted from other sections of the document, were incorporated into the training examples to help augment the marginal and conditional distributions of the tokens and their class labels. On average each document had 18 positive snippets added to it, as determined by the number of matching tokens found in our domain-specific dictionaries and gazetteers.

- Negative snippets (NEG): As described in §5.1.2, similar to POS, except examples of negatively pseudo-labeled tokens were added. On average each document had 50 negative snippets added to it, as determined by the number of matching tokens found in our domain-specific stop-lists and our mutually exclusive classes such as image pointers.

In all experiments we used the Minorthird toolkit to construct the lexical features and perform the CRF training [Cohen, 2004], and performed evaluation via cross validation over held out data (except where comparative user studies were conducted in §5.2.4 and 5.2.5, as noted).

5.2.3 Non-transfer: abstract to abstract

Table 5.1 shows the performance of seven[1] different extractors (sorted by F1), each trained on a unique combination of our proposed features: positive snippets (POS), negative snippets (NEG), and structural frequency features (FREQ), all along with the standard lexical features (LEX). A check mark in a feature's column means that row's extractor was provided with that column's features at train-time. In this non-transfer experiment, our model labeled tokens of held-out *abstracts* as protein or not, and these predictions were automatically evaluated with respect to token-level precision, recall and F1 measure using the held-out GENIA labels for those abstracts. From this table we can notice a number of trends. With respect to the baseline model (BASE) trained only on lexical features, adding positive snippets (POS) doesn't seem to help precision or recall much, while adding structural frequency features (FREQ) improves recall (and thus F1) dramatically. This makes sense, since positive snippets were proposed as a method of increasing domain-robustness, and these results are for the non-transfer setting. On the other hand, structural frequency features were proposed as a general purpose method of using an article's internal structure to help extract useful information from the unsupervised sections of the document. In this respect, FREQ features might be expected to aid in even the non-transfer setting, as they do here. Interestingly, although in isolation, and even in combination, POS and NEG snippets themselves don't seem to improve on the baseline model in the non-transfer setting, when combined with FREQ features (FULL) they do seem to provide another boost to recall. This may be due to the fact the inter-domain information implicitly incorporated by the structural frequency features allows the model to better make use of the cross-domain snippets.

We should note that, although this non-transfer, abstract to abstract setting is convenient

[1]The NEG model, containing only negative snippets, is missing, but given the results of NEG_FREQ and FREQ can be assumed to be no better than BASE.

(since we can get precise evaluation numbers) and the results encouraging, it is unclear what they might indicate about performance in the transfer setting, which we address next.

Model name	POS	NEG	FREQ	Prec	Rec	F1
FULL	✓	✓	✓	.738	.673	**.704**
FREQ			✓	.744	.640	**.688**
POS_FREQ	✓		✓	.727	.637	**.679**
POS	✓			.760	.555	.641
POS_NEG	✓	✓		.760	.547	.636
BASE				.753	.550	.636
NEG_FREQ		✓	✓	.751	.535	.625

Table 5.1: Summary of ablation study results for extractors trained on full papers and evaluated on *abstracts* (results for FREQ from Table 4.3 are included here for completeness). For F1 results, all values in **bold** are significantly greater than all those in plain font (one-sided paired t-test, $p < .01$).

5.2.4 Transfer: abstract to caption, full vs. baseline

Finally, we present the results of a user study in the domain adaptation setting. We trained extractors on various combinations of features computed on the training data, and compared them to the full model trained on lexical, structural, positive and negative snippets, evaluating each with respect to the proteins they predicted in held-out *captions*. Unlike the non-transfer setting, however, since we had no labels for any captions, we could not perform automatic evaluation. Instead, we employed human experts to manually compare the

predictions made by variously constructed extractors, side by side, and evaluate which they preferred.

Figure 5.1 shows a screenshot of the tool we used to perform these evaluations. In the top-right, two extractors are being compared: 1A in yellow and 1B in blue (their names have been blinded from the evaluator). The top-left panel shows the captions of a particular test article with each extractor's positive (protein) predictions highlighted in its color, with green highlights representing tokens on which both extractors predict positive. The bottom panel shows two columns of buttons: 1A's predictions are on the left, and 1B's on the right. Since we are evaluating user preference, only the predictions where the extractors disagree are shown. For each row (corresponding to a disagreement between extractors) the human expert clicks the cell of the prediction he prefers: clicking an empty cell in one column means the user believes the other column's extractor made a type I (false positive) error, while clicking a non-empty cell implies the other column's extractor made a type II (false negative) error.

Each of these judgments can be viewed as the outcome of a paired trial, and by using a paired t-test, we can assess how the extractors differ along with which the user prefers. Due to the nature of the hypothesis tests, however, we cannot quantify at all by *how much* the user prefers one to another, or by how much one has improved with respect to the other.

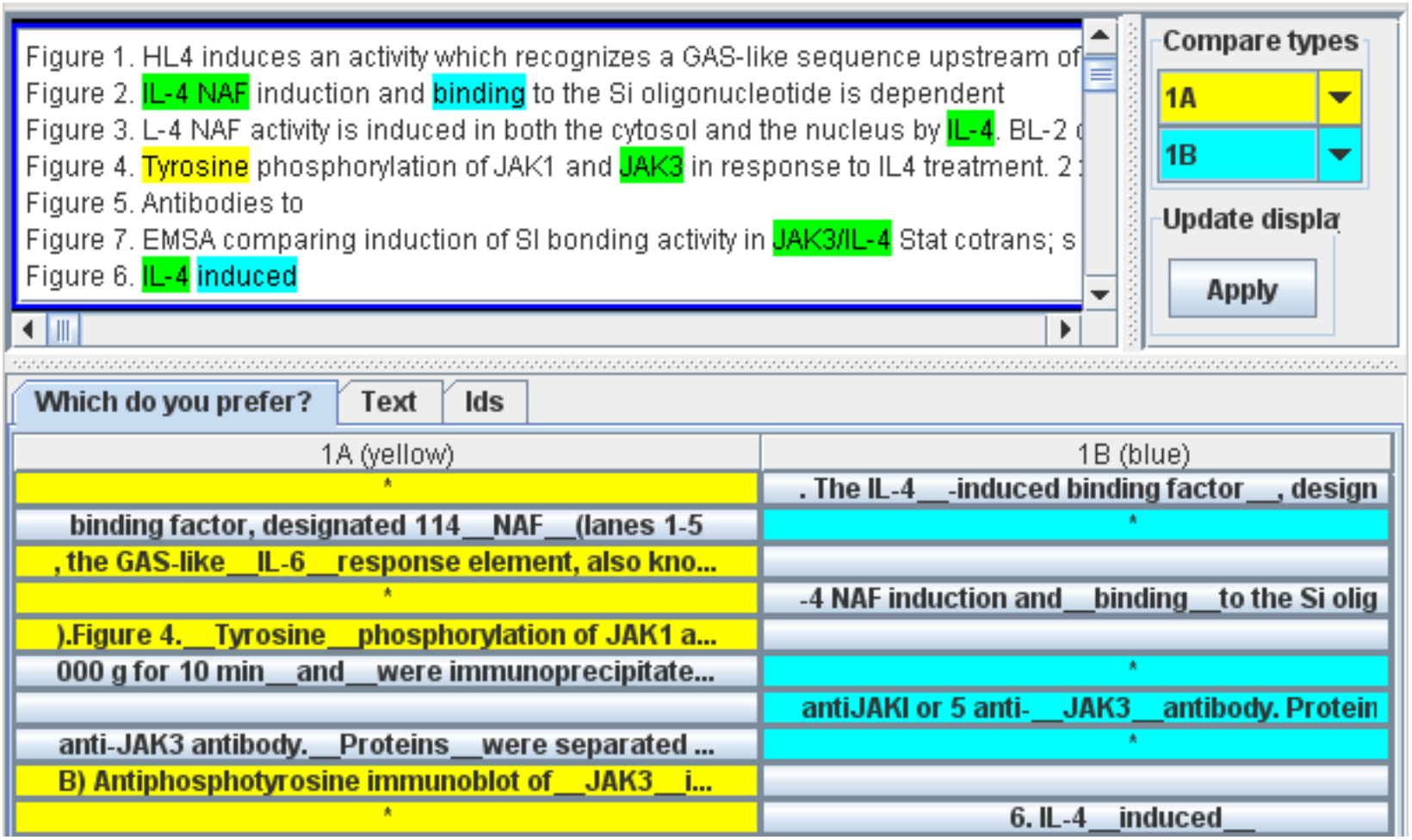


Figure 5.1: Screenshot of application used to compare various protein extractors' performance on captions in the face of no labeled data.

This makes it difficult to tell how much of a boost has been achieved by various changes to the algorithm, and puts the burden of thoughtful experiment design on the researcher in order to test instructive hypotheses. Another downside to the user-evaluation approach is that it requires a new study to be performed after every change to the algorithm, thus encouraging well-planned, if frugal, iterations.

This issue of evaluation in the absence of labeled test data is not unique to our experiments, however, and is endemic to all types of unsupervised, semi-supervised and transfer learning problems. The issues is that, in these learning settings, data from the test domain is by definition scarce or non-existent. Even when there is some labeled test data present, it is usually far preferable to use what is available for training, rather than reserve it for evaluation. Thus it is necessary to come up with evaluation methods, such as our comparative user study, that make do without labeled test data.

Although pre-labelled test datasets provide a convenient benchmark against which to perform repeated, automated evaluations, they are expensive. We found the expense of performing side by side hand-evaluations to be relatively low (given a thoughtful experiment design and user-interface). They also have the added benefit of being robust to issues such as inter-annotator agreement, which can plague a highly technical domain such as biological entity tagging. For example, while two expert annotators may not agree on the precise boundaries of a complex protein entity span (and thus cause confusion for the learner if one expert labeled the training data, and another the test data), they are more likely to have consistent standards when comparing the proposed methods during test time, and thus provide consistent results that may be aggregated, reducing the number of comparisons needed to reach consensus. This user-preference based method is also more efficient than comparing fully-annotated articles if the various classifiers being compared frequently agree, since human effort will only be spent comparing differences, rather than labeling large stretches of

identical predictions.

Using our user study method we found that our proposed model (FULL, the joint combination of all three new feature types: POS, NEG and FREQ) was preferred by users significantly more often ($p < .01$, see Table 5.2) than the baseline model trained only on lexical features.

Evaluation is an important consideration in semi-supervised domain adaptation, since, by definition, no labeled test (target domain) data is available. The type of comparative evaluation we performed could be instrumented into various end-user applications (for example, click-through logs from protein name search engines such as SLIF[2]) to automatically extract the necessary user-preference information, thus obviating the need of a special evaluator.

5.2.5 Transfer: abstract to caption, full vs. ablated

Having established that a model based on a combination of our new features (incorporated in the FULL model) improved user preference over the baseline, purely lexical model, we then performed an ablation study to ascertain which of these new features (structural frequency (FREQ), positive snippets (POS), or negative snippets (NEG)) were responsible for the improvements observed. Table 5.2 summarizes these results for each ablation considered. In each such study comparing the full model to a degraded model, the full model was preferred significantly more often than the ablated model (one-sided paired t-test, $p<.01$), indicating that our proposed features are, in fact, useful for unsupervised domain adaptation.

In addition, it should be noted that, although the lack of labeled target data required us to use user studies to compare methods, we were able to reach high-confidence conclusions after only a relatively small number of hand-evaluations, due to the statistical efficiency of our paired tests. This should lend encouragement to those hesitant to tackle problems lacking

[2]http://slif.cbi.cmu.edu/

labeled test data, for fear of tedious hand-labeled evaluations.

Preferred model	Compared to	p-value	# user labels	Equivalent # documents
FULL	BASE	3.6 E-4	182	3.64
FULL	NEG_FREQ	9.9 E-9	78	1.56
FULL	POS_NEG	1.8 E-4	120	2.40
FULL	POS_FREQ	1.1 E-4	46	.92

Table 5.2: Summary of transfer results for extractors trained on full papers and evaluated on *captions*. The preferred model is in bold. *Equivalent # documents* is calculated by comparing the number of user labels required in our side by side evaluation to those needed by an automated system, requiring a fully-annotated document (in this case, an image *caption*), with about 50 labeled tokens per document.

From these results we can further observe that adding POS snippets seems to have a noticeable effect on user preference (since FULL is prefered to NEG_FREQ). This is a nice complement to the result from §5.2.3 which indicated that POS snippets are not as useful in the non-transfer setting. Indeed, it is the ability of POS snippets to shape the labeled training source data to look more like the target data that allows the extractors so trained to be robust across shifts in domains. Similar user preference is seen for the contribution of NEG snippets and FREQ features, indicating that they too aid in domain-adaptation, both by leveraging unlabeled training data and by helping to inform the training data with some target domain attributes.

5.3 Conclusions: snippets & structural frequency features

In these chapters we have shown how exploiting structure, in the form of frequency features and positive and negative snippets, can help produce robust extractors that overcome the problem of semi-supervised domain adaptation. We have defined a new set of features based on structural frequency statistics and demonstrated their utility in representing inter-domain information drawn from both supervised and unsupervised sources, in a manner somewhat orthogonal to the traditional lexically based feature sets. Towards a similar goal of robust cross-domain learning, we have defined a technique for introducing high-confidence positively and negatively labeled pseudo examples (snippets) from the target domain into the source domain, and shown that these too provide a convenient, and effective, method for producing an extractor that is robust to domain shifts between training and testing data sets. Finally, through a comparative analysis of each new feature's contribution to same-domain and inter-domain information extraction performance, we have discovered an intriguing relationship between a feature's utility in the non-transfer and transfer settings. Along the way, in order to assess our transfer techniques' performance in the face of a lack of labeled test data, we have also developed a novel framework for human evaluation that facilitates statistically interpretable paired testing.

Chapter 6

Graph Relations for Robust Named Entity Recognition

Recall that our goal throughout this thesis has been to discover patterns within and relationships among various sources of data, and to investigate and exploit these regularities in order to produce learners that are more robust across shifts in data and task. More abstractly, Figure 1.1 shows how each learning problem can be represented as a tuple $(X, Y, \ldots Z)$ of features, labels and other domain and task specific *metadata* such as the *feature hierarchies* relating source-domain features to target-domain features in Chapter 3, or the *structural frequency features* and *snippets*' one-sense-per-discourse assumption relating source-domain tokens to target-domain tokens in Chapters 4 and 5.

In this chapter we use the problem of relating tokens in source abstracts to tokens in related target abstracts in the biomedical literature as a motivating example with which to demonstrate how we can explicitly model these types of metadata-derived relationships as edges and paths in a general graph. Although we focus in this chapter on citation-based metadata such as *authorship* and *citation*, our graph representation should be flexible enough

to express most other types of metadata and thus should be applicable to many other new problems.

We break down the problem of relating tokens across abstracts into two phases:

- Section 6.1 establishes that meaningful relationships hold between *author* and *gene* entities. This is verified by link prediction experiments.
- Section 6.2 establishes that similar relationships help for *in-task* generalization for NER (just as *structural frequency features* and *snippets* were shown to do).

We leave the analogous *cross-task* experiments, which would examine these methods' effectiveness in transferring from abstracts to captions and across biological subdomains, for future work.

The rest of the chapter is organized as follows: Section 6.1 begins with an introduction to the idea of *annotated citation networks* in §6.1.1 while §6.1.2 provides details of their implementation and construction. §6.1.3 discusses our *graph-walk* based method of extracting useful information from these networks, while §6.1.4 relates how we used this method on our data to help predict which genes an author would write about in the future. The results of these experiments, along with concluding remarks and related work, are summarized in Sections 6.1.5 and 6.1.6.

Section 6.2 is organized in a parallel fashion: §6.2.1 relates our success at predicting genes from authors using citation networks (in §6.1) to the more central problem of robust named entity recognition. §6.2.2 recalls the data used for these *graph-based* NER experiments (almost identical to those of §6.1.2), while §6.2.3 describes our method for combining graph-based predictions with standard lexical features to create *graph-augmented named entity extractors*. These augmented extractors are then compared to standard lexically trained ones in §6.2.4, with the results detailed and summarized in Sections 6.2.5 and 6.2.6

6.1 Graph relations for cross-task learning

We demonstrate the usefulness of various types of publication-related metadata, such as citation networks and curated databases, for the task of identifying genes in academic biomedical publications. Specifically, we examine whether knowing something about which genes an author has previously written about, combined with information about previous coauthors and citations, can help us predict which new genes the author is likely to write about in the future [Arnold and Cohen, 2009]. Framed in this way, the problem becomes one of predicting links between authors and genes in the publication network. We show that this social-network based link prediction technique outperforms various baselines, including those relying only on non-social biological information, suggesting a fruitful combination with already present lexical information to create more robust named entity extractors (further explored in Section 6.2).

6.1.1 Introduction

Although academics have long recognized and investigated the importance of citation networks, their investigations have often been focused on historical [Garfield et al., 1964], summary, or explanatory purposes [Erosheva et al., 2004; Liu et al., 2005; Cardillo et al., 2006; Leicht et al., 2007]. While other work has been concerned with understanding how influence develops and flows through these networks [Dietz et al., 2007], we instead focus on the problem of link prediction [Cohn and Hofmann, 2001; Liben-Nowell and Kleinberg., 2003]. *Link prediction* is the problem of predicting which nodes in a graph, currently unlinked, 'should' be linked to each other, where 'should' is defined in some application-specific way. This may be useful to know if a graph is changing over time (as in citation networks when new papers are published), or if certain edges may be hidden from observation (as in detecting

insider trading cabals). In our setting, we seek to discover edges between authors and genes, indicating genes about which an author has yet to write, but which he may be interested in.

We define a *citation network* as a graph in which *publications* and *authors* are represented as nodes, with bi-directional *authorship* edges linking authors and papers, and uni-directional *citation* edges linking papers to other papers (the direction of the edge denoting which paper is doing the citing and which is being cited). We can construct such a network from a given corpus of publications along with their lists of cited works. There exist many so called *curated* literature databases for biology in which publications are *tagged*, or manually labeled, with the genes with which they are concerned. We can use this metadata to introduce *gene* nodes to our enhanced citation network, which are bi-directionally linked to the papers in which they are tagged. Finally, we exploit a third source of data, namely biological domain expertise in the form of ontologies and databases of facts concerning these genes, to create *association* edges between genes which have been shown to relate to each other in various ways. We call the entire structure an *annotated citation network*.

In the following subsections, respectively, we discuss the topology of our annotated citation network, along with describing the data sources from which the network was constructed. We then employ *random walks*, a technique used for calculating the proximity of nodes in our graph, thus suggesting plausible novel links between authors and genes. Finally, we describe an extensive set of ablation studies performed to assess the relative importance of each type of edge, or *relation*, in our model and discuss the results, concluding with a view towards a future model combining network and text information in Section 6.2.

6.1.2 Data

We are lucky to have access to many sources of high quality data:

- PubMed and PubMed Central (PMC): PubMed is a free, open-access on-line archive of over 18 million biological abstracts and bibliographies, including citation lists, for papers published since 1948 [U.S. National Library of Medicine, 2008]. PubMed Central contains full-text copies of over one million of these papers for which open-access has been granted [National Institues of Health, 2009].

- The Saccharomyces Genome Database (SGD): A database of various types of information concerning the yeast organism *Saccharomyces cerevisiae*, including descriptions of its genes along with over 40,000 papers manually tagged with the genes they mention [Dwight et al., 2004].

- The Gene Ontology (GO): A large ontology describing the properties of and relationships between various biological entities across numerous organisms [Consortium, 2000].

From the data provided by these sources we are able to extract the nodes and edges that make up our annotated citation network, shown graphically in Figure 6.1. Specifically our network consists of the following.

Nodes

The nodes of our network represent the *entities* we are interested in.

- 44,012 *Papers* contained in SGD for which PMC bibliographic data is available.

- 66,977 *Authors* of those papers, parsed from the PMC citation data. Each author's position in the paper's citation (i.e. first author, last author, etc.) is also recorded, although it is not represented in the graph.

- 5,816 *Genes* of yeast, mentioned in those papers.

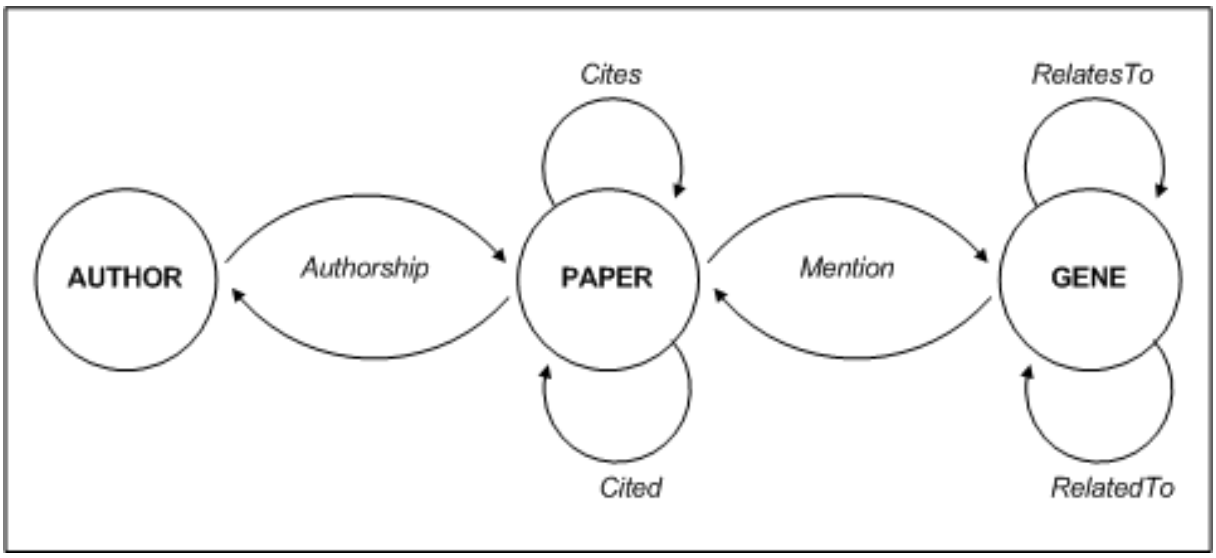


Figure 6.1: Topology of the full annotated citation network, node names are in bold while edge names are in italics.

Edges

We likewise use the edges of our network to represent the *relationships* between and among the nodes, or entities.

- *Authorship*: 178,233 bi-directional edges linking author nodes and the nodes of the papers they authored.
- *Mention*: 160,621 bi-directional edges linking paper nodes and the genes they discuss.
- *Cites*: 42,958 uni-directional edges linking nodes of citing papers to the nodes of the papers they cite.
- *Cited*: 42,958 uni-directional edges linking nodes of cited papers to the nodes of the papers that cite them
- *RelatesTo*: 1,604 uni-directional edges linking gene nodes to the nodes of other genes

appearing in their GO description.

- *RelatedTo*: 1,604 uni-directional edges linking gene nodes to the nodes of other genes in whose GO description they appear.

The SGD database contains papers published from 1950 through 2008, with the number of papers annotated growing exponentially each year, as shown in Figure 6.2. The relationships between genes, derived from GO, are likewise labeled with the year in which they were discovered. This allows us to conveniently segment all the data chronologically, enabling pure temporal cross validation[1].

6.1.3 Methods

Now that we have a representation of the data as a graph, we are ready to begin the calculation of our link predictions. To motivate our algorithm, imagine the first step is to pick a node, or set of nodes, in the graph to which our predicted links will connect. These are our *query nodes*. We then perform a *random walk* out from the query node, simultaneously following each edge to the adjacent nodes with a probability proportional to the inverse of the total number of adjacent nodes [Cohen and Minkov, 2006]. We repeat this process a number of times, each time spreading our probability of being on any particular node, given we began on the query node. If there are multiple nodes in the query set, we perform our walk simultaneously from each one. After each step in our walk we have a probability distribution over all the nodes of the graph, representing the likelihood of a walker, beginning at the query node(s) and randomly following outbound edges in the way described, of being on that particular node. Under the right conditions, after enough steps this distribution will converge (a full discussion of the criteria and rates of convergence for random walks is beyond

[1]An on-line demo of our work, including links to the network data file used for the experiments, can be found at http://yeast.ml.cmu.edu/nies/.

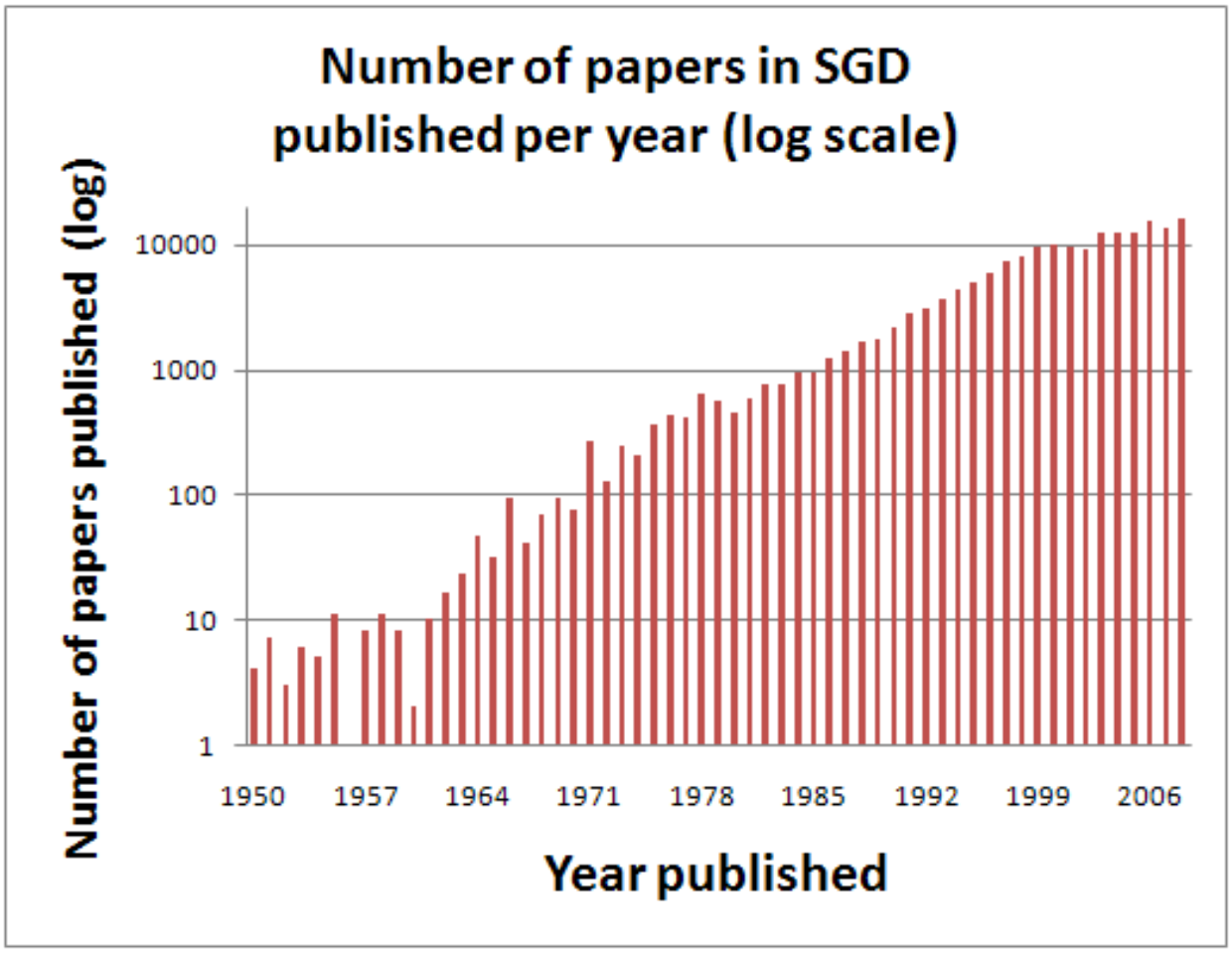


Figure 6.2: Distribution of papers published per year in the SGD database.

the scope of this thesis, but suffice it to say that there are a wide variety of variations of the simple random walk technique that can deal with most degenerate cases). We can then use this distribution to rank all the nodes, predicting that the nodes most likely to appear in the walk are also the nodes to which the query node(s) should most likely connect.

We interpret the fact that there are more (weighted) paths from a given author to a given gene as suggesting that the query author is more likely to write about the predicted gene in the future. We feel comfortable making this interpretation since the only edge-type joining an *Author node* to a *Gene node* that we have modeled in our training network is the *Authorship* relation. Thus, when a similar coupling is predicted by the graph walk, we interpret the predicted edge as suggesting that the query author is likely to write about the predicted gene in the future, just as the analogous edge in the training data represented the fact that an author wrote about a gene in the past. This interpretation seems safe to make in cases where there are constrained semantics for edge-types joining certain classes of nodes. If there were multiple similarly typed edges (for instance, $Author \rightarrow Gene$ edges representing an author's disdain for a gene) the results of the random walk would be more ambiguous.

In practice, the same results can be achieved by multiplying the adjacency matrix of the graph by a vector representing the current distribution over the graph, that is, the probability of being on any one node. This adjacency matrix may be weighted to reflect the varying strength of different edge types, as well as the fact that transition probabilities are normalized over all out-edges from a node. Each such multiplication represents one complete step in the walk, resulting in an updated distribution over the nodes of the graph.

We can adjust the adjacency matrix (and thus the graph) by selectively hiding, or removing, certain *types* of edges. For instance, if we want to isolate the influence of citations on our walk, we can remove all the citation edges from the graph, perform a walk, and compare the results to a walk performed over the full graph.

Likewise, in order to evaluate our predicted edges, we can hide certain instances of edges, perform a walk, and compare the predicted edges to the actual withheld ones. For example, if we have all of an author's publications and their associated gene mention data for the years 2007 and 2008, we can remove the links between the author and the genes he mentioned in 2008 (along with all other edges gleaned from 2008 data), perform a walk, and then see how many of those withheld gene-mention edges were correctly predicted. Since this evaluation is a comparison between one unranked set (the true edges) and another ranked list (the predicted edges) we can use the standard information retrieval metrics of precision, recall and F1.

6.1.4 Experiment

To evaluate our network model, we first divide our data into two sets:

- `Train`, which contains only *authors*, *papers*, *genes* and their respective relations which were published before 2008
- `Validation`, which contains new[2] ($author \overset{Mentions}{\rightarrow} genes$) relationships that were first published in 2008.

From this `Train` data we create a series of subgraphs, each emphasizing a different set of relationships between the nodes. These subgraphs are summarized in Figure 6.3. By selectively removing edges of a certain type from the $FULL$ graph we were able to isolate the effects of these relations on the random walk and, ultimately, the predicted links. Specifically, we classify each graph into one of four groups and later use this categorization to asses the

[2]We restrict our evaluation to genes about which the author has never previously published (even though an author may publish about them again in 2008), since realistically, these predictions would be of no value to an author who is already familiar with his own previous publications.

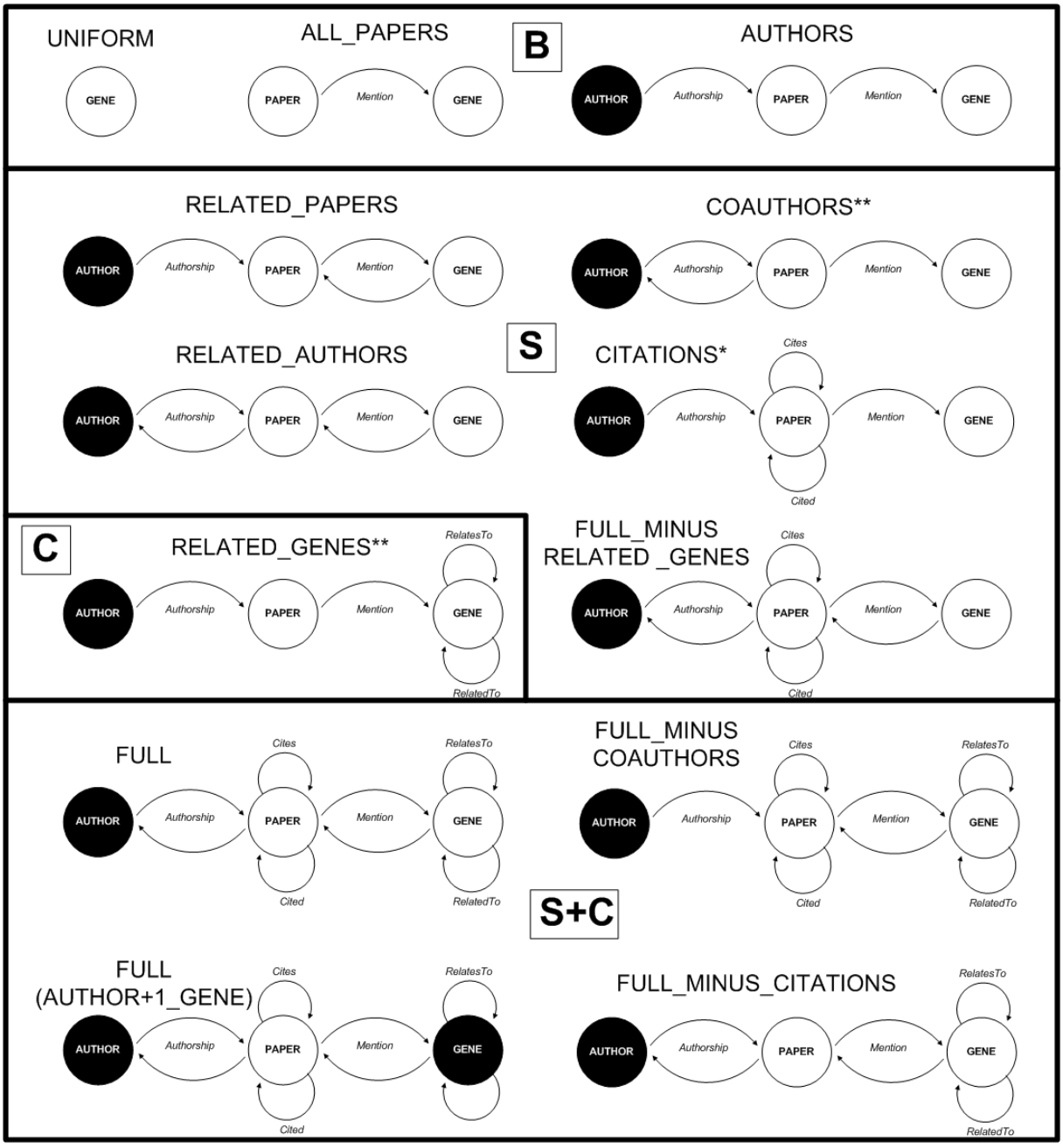


Figure 6.3: Subgraphs queried in the experiment, grouped by type: **B** for baselines, **S** for social networks, **C** for networks conveying biological content, and **S+C** for networks making use of both social and biological information. Shaded nodes represent the node(s) used as a query. **For graph $RELATED\_GENES$, which contains the two complimentary uni-directional *Relation* edges, we also performed experiments on the two subgraphs $RELATED\_GENES_{RelatesTo}$ and $RELATED\_GENES_{RelatedTo}$ which each contain only one direction of the *relation* edges. For graph $CITATIONS$, we similarly constructed subgraphs $CITATIONS_{Cites}$ and $CITATIONS_{Cited}$.

relative contribution of each edge type to the overall link prediction performance.

Baseline

The baseline graphs are $UNIFORM$, ALL_PAPERS and $AUTHORS$. $UNIFORM$ and ALL_PAPERS do not depend on the *author* node. $UNIFORM$, as its name implies, is simply the chance of predicting a novel gene correctly given that you select a predicted gene uniformly at random from the universe of genes. Since there are 5,816 gene names, and on average each author in our query set writes about 6.7 new genes in 2008, the chance of randomly guessing one of these correctly is $6.7/5816 = .12\%$. Using these values we can extrapolate this model's expected precision, recall and F1. Relatedly, ALL_PAPERS, while also independent of authors, nevertheless takes into account the distribution of genes across papers in the training graph. Thus its predictions are weighted by the number of times a gene was written about in the past. This model provides a more reasonable baseline. $AUTHORS$ considers the distribution of genes over all papers previously published by the author. While this type of model may help recover previously published genes, it may not do as well identifying new genes.

Social

The social graphs ($RELATED\_PAPERS$, $RELATED\_AUTHORS$, $COAUTHORS$, $FULL\_MINUS\_RELATED\_GENES$ and $CITATIONS$) are constructed of edges that convey information about the social interactions of authors, papers and genes. These include facts about which authors have written together, which papers have cited each other, and which genes have been mentioned in which papers.

Content

In addition to social edges, some graphs also encode information regarding the biological content of the genes being published. The graph $RELATED\_GENES$ models only this biological content, while $FULL\_MINUS\_COAUTHORS$, $FULL\_MINUS\_CITATIONS$, $FULL$ and $FULL(AUTHOR + 1\_GENE)$ all contain edges representing both social and biological content.

Protocol

For our query nodes we select the subset of authors who have publications in both the `Train` and `Validation` set. To make sure we have fresh, relevant publications for these query authors, and to minimize the impact of possible ambiguous name collision, we further restrict the query author list to only those authors who have publications in both 2007 and 2008. This yields a query list, AllAuthors, with a total of 2,322 authors, each to be queried independently, one at a time. We further create two other query author lists, FirstAuthors and LastAuthors containing 544 and 786 authors respectively, restricted to those authors who appear as the first or last author, respectively, in their publications in the `Validation` set. The purpose of these lists of queries is to determine whether an author's position in a paper's list of authors has any impact in our ability to predict the genes he or she might be interested in.

Given these sets of graphs and query lists, we then query each author in each of our three lists, independently, against each subgraph in Figure 6.3. Each such (author, graph) query yields a ranked list of genes predicted for that author given that network representation. By comparing this list of predicted genes against the set of true genes from `Validation` (i.e. the new genes query authors published about in the held-out 2008 publication data) we are

able to calculate the performance of each (author, graph) pairing[3]. These resulting precision, recall, F1 and MAP metrics, broken down for each set of author positions, are summarized in Figure 6.5 respectively.

Querying with extra information

Finally, we were interested in seeing what effect adding some limited information about an author's 2008 publications to our query would have on the quality of our predictions. This might occur, for instance, if we have the text of one of the author's new papers available and are able to perform basic information extraction to find at least one gene. The question is, can we leverage this single, perhaps easy to identify gene, to improve our chances of predicting or identifying other undiscovered new genes? To answer this question, in addition to querying each author in isolation, we also queried, together as a set, each author and the one new gene about which he published most in 2008 (see graph $FULL(AUTHOR + 1\_GENE)$ in Figure6.3). These results are summarized, along with the others, in Figure 6.5, again broken down by author position.

6.1.5 Results

Using Figures 6.3, 6.4 and 6.5 as guides, we turn now to an analysis of the effects different edge types have on our ability to successfully predict new genes. We should first explain the absence of results for the $AUTHORS$ graph, and the lines for $UNIFORM$ and ALL_PAPERS in Figures 6.4 and 6.5. Since these baselines do not depend on the query, they are constant across models and are thus displayed as horizontal lines across the charts in Figures 6.4 and 6.5. $AUTHORS$ is missing because it is only able to discover genes

[3]Since the list of predicted genes is sometimes quite long (since it is a distribution over all genes in the walk), we set a threshold and all evaluations are calculated only considering the top 20 predictions made (in practice, this choice of threshold did not affect the relative performance of the models much).

that have *already* been written about by the query authors in the training graph. Since our evaluation metrics only count the prediction of novel genes, $AUTHORS$'s performance is necessarily zero.

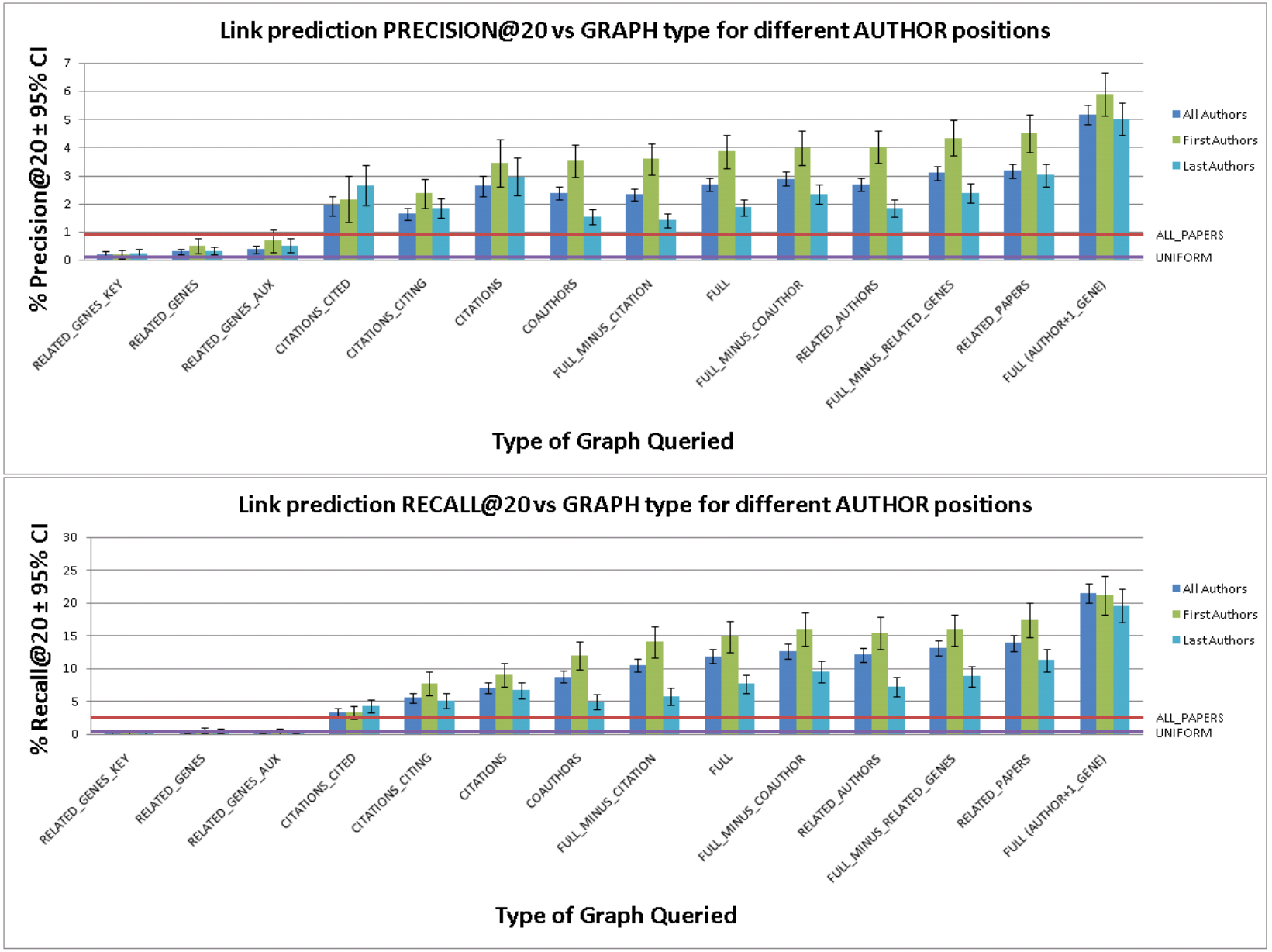


Figure 6.4: Mean percent precision and recall @20 of queries across graph types, broken down by author position, shown with error bars demarking the 95% confidence interval. Baselines $UNIFORM$ and ALL_PAPERS are also displayed.

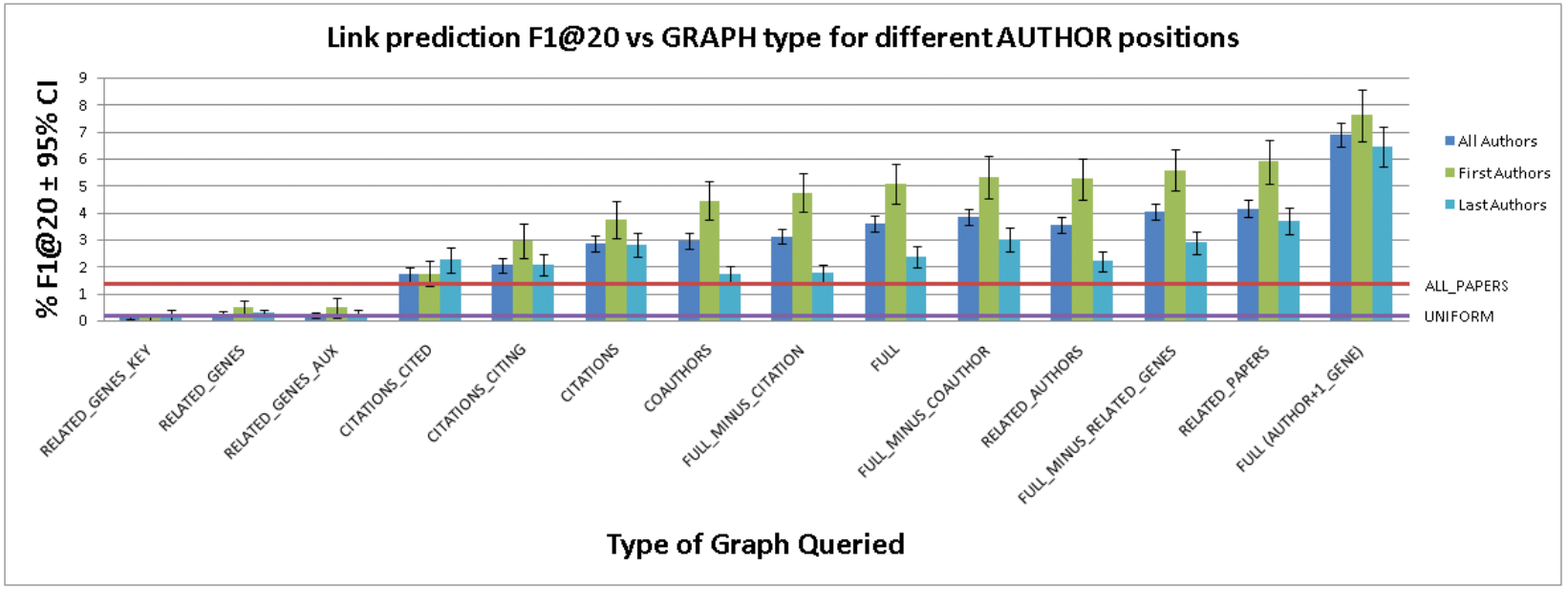


Figure 6.5: Mean percent F1 @20 of queries across graph types, broken down by author position, shown with error bars demarking the 95% confidence interval. Baselines $UNIFORM$ and ALL_PAPERS are also displayed.

Given these baselines, let us next consider the role of author position on prediction performance. It is apparent from the results that, in almost all settings, querying based on the first author of a paper generates the best results, with querying by last author performing the worst. This seems to suggest that knowing the first author of a paper is more informative than knowing who the last author was in terms of predicting which genes that paper may be concerned with. Depending on the specifics of one's own discipline, this may be surprising. For example, in computer science it is often customary for an advisor, lab director or principal investigator to be listed as the last author. One might assume that the subject of that lab's study would be most highly correlated with this final position author, but the evidence here seems to suggest otherwise. Tellingly, the only case in which the last author *is* most significant[4] is in the $CITATIONS\_CITED$ model. Recall that in this model only edges from cited papers to their citing papers are present. These results may suggest that in this model, knowing the last author of the paper actually is more valuable. This might be explained by the assumption that the actual scientific content of an article is best indicated by the primary person conducting the experiment, who in this field is usually the first author. When it comes time to create a bibliography, however, the citer may be more likely to remember related work with respect to the more senior member of the research team (in this domain, usually the last author), within whose general research area the specific work lies.

Given that in most cases the models queried using first authors performed the best, the columns of Figures 6.4 and 6.5 have been positioned in order of increasing first author F1 performance, and all subsequent comparisons are made with respect to the first author queries, unless otherwise stated. Thus we notice that those models relying solely on the biological GO information relating genes to one another (**Content** graphs from Figure 6.3)

[4]Measured by 80% confidence intervals.

perform significantly[5] worse than any other model, and are in fact in the same range[6] as the $UNIFORM$ model. Indeed, the $FULL$ model benefits from having the relations removed, as it is outperformed[5] by the $FULL\_MINUS\_RELATED\_GENES$ model.

There are a few possible explanations for why these content-based biological edges might be hurting performance. First, scientists might not be driven to study genes which *have already been demonstrated* to be biologically related to one another. Since we are necessarily using biological facts already discovered, we may be behind the wave of new investigation. Second, these new investigations, some of them biologically motivated, might not always turn out conclusively or successfully. This would likewise lead to the genes being studied in this way lying outside the scope of our biological content. Finally, it is possible that our methods for parsing and interpreting the GO information and extracting the relationships between genes may not be capturing the relevant information in the same way a trained biologist might be able to. Relatedly, the ontologies themselves might be designed more for summarizing the current state of knowledge, rather than suggesting promising areas of pursuit.

In contrast, the models exploiting the **social** relationships in $CITATIONS$, $COAUTHORS$, $RELATED\_AUTHORS$ and $RELATED\_PAPERS$ all outperform[7] the ALL_PAPERS baseline. While each of these social edge types is helpful on its own, their full combination is, perhaps counter-intuitively, not the best performing model. Indeed, while $FULL$ outperforms[5] its constituent $CITATIONS$ and $COAUTHORS$ models, it nevertheless benefits slightly[8] from having the *coauthor* edges removed (as in $FULL\_MINUS\_COAUTHOR$). This may be due to competition among the edges for the probability being distributed by our random walk. The more paths there are out of a node, the less likely the walker is to follow any given one. Thus, by removing the (many) coauthorship edges from the $FULL$

[5] $p < .01$ using the Wilcoxon signed rank test.

[6] Containing the $UNIFORM$ baseline in their 95% confidence intervals.

[7] Baseline is out of their 95% confidence intervals.

[8] $p < .15$ using the Wilcoxon signed rank test.

graph, we allow the walk to reach a better solution more quickly.

Interestingly, the best performance[9] of the single-author query models is achieved by the relatively simple, pure collaborative filtering $RELATED\_PAPERS$ model [Goldberg et al., 1992]. Explained in words, this social model predicts that authors are likely to write about genes that co-occur with an author's previously studied genes in other people's papers. This makes sense since, if other people are writing about the same genes as the author, they are more likely to share other common interests and thus would be the closest examples of what the author may eventually become interested in in the future.

Finally we examine the question of whether having not only a known author to query, but also one of this author's new genes, aids in prediction. The results for the $FULL(AUTHOR + 1\_GENE)$ model[10] seem to indicate that the answer is yes. Adding a single known new gene to our author query of the $FULL$ model improves our prediction performance by almost 50%, and significantly outperforms[11] the best single-author query model, $RELATED\_PAPERS$, as well. This is a promising result, as it suggests that the information contained in our network representation can be combined with other sources of data (gleaned from performing information extraction on papers' text, for example) to achieve even better results than either method alone.

6.1.6 Related work & Conclusions

While there has been extensive work on analyzing and exploiting the structure of networks such as the web and citation networks [Kleinberg, 1999; Kleinberg et al., 1999], most of the techniques used for identifying and extracting biological entities directly from publication

[9] $p < .10$ using the Wilcoxon signed rank test.

[10] During evaluation the queried new gene is added to the set of previously observed genes and thus does not count towards precision or recall.

[11] $p < .02$ using a paired sign test.

text [Cohen and Hersh, 2005; Feldman et al., 2003; Murphy et al., 2004; Franzén et al., 2002; Bunescu et al., 2004; Shi and Campagne, 2005] and curated databases [Wang et al., 2008] rely on performing named entity recognition on the text itself [Collins and Singer, 1999] and ignore the underlying network structure entirely. While these techniques perform well given a paper to analyze, they are impossible to use when such text is unavailable, as in our link prediction task.

In this section we have introduced a new graph-based annotated citation network model to represent various sources of information regarding publications in the biological domain. We have shown that this network representation alone, without any features drawn from text, is able to outperform competitive baselines. Using extensive ablation studies we have investigated the relative impact of each of the different types of information encoded in the network, showing that social knowledge often trumps biological content, and demonstrated a powerful tool for both combining and isolating disparate sources of information.

We have further shown that, in the domain of *Saccharomyces* research from which our corpus was drawn, knowing who the first author of a paper is tends to be more informative than knowing who the last author is (contrary to some conventional wisdom). We have also shown that, despite performing well on its own, our network representation can easily be further enhanced by including in the query set other sources of knowledge about a prediction subject gleaned from separate techniques, such as information extraction and document classification.

With respect to **same domain multi-task transfer**, we have shown that we can use instances and labels across various tasks (such as paper_ids labeled with authors, citations and genes) to help predict future authors and genes. Relatedly, we have shown that it is easier to perform $author \Rightarrow gene$ prediction if we also have $author \Rightarrow paper$, $paper \Rightarrow gene$ and $paper \Rightarrow paper$ relations. We show $gene \Rightarrow gene$ relations are not helpful.

Finally, we have shown that **external data sources** such as citation networks (PMC), gene ontologies (GO) and curated databases (SGD) can be combined to form curated citation networks which can be exploited to improve *author* $\Rightarrow$ *gene* prediction, and that a **limited and well studied domain**, such as yeast as represented in SGD, provides an ideal test-bed for quickly developing and evaluating novel robust learning techniques. The key features that allow this are large amounts of different kinds of relatively noise free data (such as curated databases, citation lists and gene ontologies) giving different views of the problem domain, and, crucially, some normalized representation of entities across those data sources (PubMed ID's, author names and gene identifiers) allowing one to join facts between them.

6.2 Graph-based priors for named entity extraction

6.2.1 Introduction & goal

Given the success of the curated citation networks of Section 6.1 in *predicting* which genes an author might write about in the future, along with our underlying goal of discovering and exploiting interesting relationships between various aspects of data and tasks to produce more robust learners, this section demonstrates how we are able to exploit this same network-based information, combined with common lexical features, into a CRF-based extractor for robustly *recognizing* genes in text.

6.2.2 Data

For this combined experiment, since it required both labeled abstracts and a curated citation network, we used the intersection of the data from §4.2 and §6.1.2, namely, 298 GENIA abstracts for which PMC, SGD and GO information was available, along with protein la-

bels. We split this data into training and testing splits, and built citation networks for each split (*train_citation_network*, *test_citation_network*), along with a combined network (*combined_network*).

6.2.3 Method

During training, each abstract is presented to MinorThird to have its tokens' features constructed and evaluated. At the start of this process, each abstract's set of authors are queried against a curated citation network, and a ranked list of predicted genes is returned. This ranked list of genes is then broken down into five constituent dictionaries comprised of the top 5, 10, 20, 50 and 100 results each. These dictionaries of the top-K predicted genes for the author set of each training abstract are then added to MinorThird's definition of features (as explained in Appendix A). Thus, each token in the given abstract, in addition to the normal set of lexical features, is tagged with features describing whether it is a member of each of the top-K lists. All these features, for each token within each abstract in the training data, are then presented to the CRF model to be learned.

Once a model has been trained, predictions are made on the held-out test data in an analogous way: each test abstract is queried against the *test_citation_network*, a ranked list of genes is returned and turned into a set of features, each of which is evaluated for all tokens of that abstract. This complete feature vector is then passed to the trained CRF and a prediction is made. These predictions are aggregated in the normal way.

These train and prediction methods are summarized in Tables 6.1 and 6.2 respectively.

Input: $Abstract$ = Labeled abstract to be trained upon
$Citations$ = Citation network
$Thresholds$ = Threshold for predicting a gene
Train: $Authors = ExtractAuthors(Abstract)$
$RankedGenes = RankGenes(Authors, Citations)$
$PredictedGenes = PredictGenes(RankedGenes, Thresholds)$
$Features = LexcialFeatures(Abstract, PredictedGenes)$
$CRF = TrainCRF(Features)$
Output: CRF

Table 6.1: Algorithm for training a model built upon graph-based priors over lexical features.

Input: $Abstract$ = Test abstract to be labeled
$Citations$ = Citation network
$Thresholds$ = Threshold for predicting a gene
CRF = Model trained using graph-based priors
Prediction: $Authors = ExtractAuthors(Abstract)$
$RankedGenes = RankGenes(Authors, Citations)$
$PredictedGenes = PredictGenes(RankedGenes, Thresholds)$
$Features = LexcialFeatures(Abstract, PredictedGenes)$
$PredictedGenes = PredictGenes(Features, CRF)$
Output: $PredictedGenes$

Table 6.2: Algorithm for predicting using a model built upon graph-based priors over lexical features.

6.2.4 Experiment

We performed three experiments to evaluate the contribution of the curated citation network features to our standard lexical-feature-based CRF extractor:

- **CRF_LEX**: The standard CRF model trained on the standard lexical features described in §4.1.1 (LEX).
- **CRF_LEX+GRAPH_SUPERVISED**: The standard CRF model trained on the standard lexical features, augmented with curated citation network based features (GRAPH). In this GRAPH_SUPERVISED model, training data abstracts were queried against the *train_citation_network* comprised solely of citation data concerning the papers in the training corpus.
- **CRF_LEX+GRAPH_TRANSDUCTIVE**: Similar to the *CRF_LEX + GRAPH_SUPERVISED* model, except, during training, instead of querying the *train_citation_network*, this model queries the *combined_network*, comprised of citation data concerning both the train and test papers, but with all the gene nodes and edges from the test papers to gene nodes removed. This type of semi-supervised training is possible since no textual data or class labels are needed or used during the citation network graph walk, only the structure of the citation network itself is utilized. This method is labeled 'TRANSDUCTIVE' since it attempts to take advantage of the unlabeled structure of the test data (in this case, its citation network) during training.

6.2.5 Results

The results of these experiments are summarized in Figure 6.6. We can clearly see that the addition of the citation network graph walk based features (*CRF_LEX+GRAPH_SUPERVISED*

and *CRF_LEX+GRAPH_TRANSDUCTIVE*) improves extractor performance over the purely lexical based baseline (*CRF_LEX*). We do not, however, see a significant difference in performance between the supervised and transductive versions of the augmented features (*CRF_LEX+GRAPH_SUPERVISED* vs. *CRF_LEX+GRAPH_TRANSDUCTIVE*).

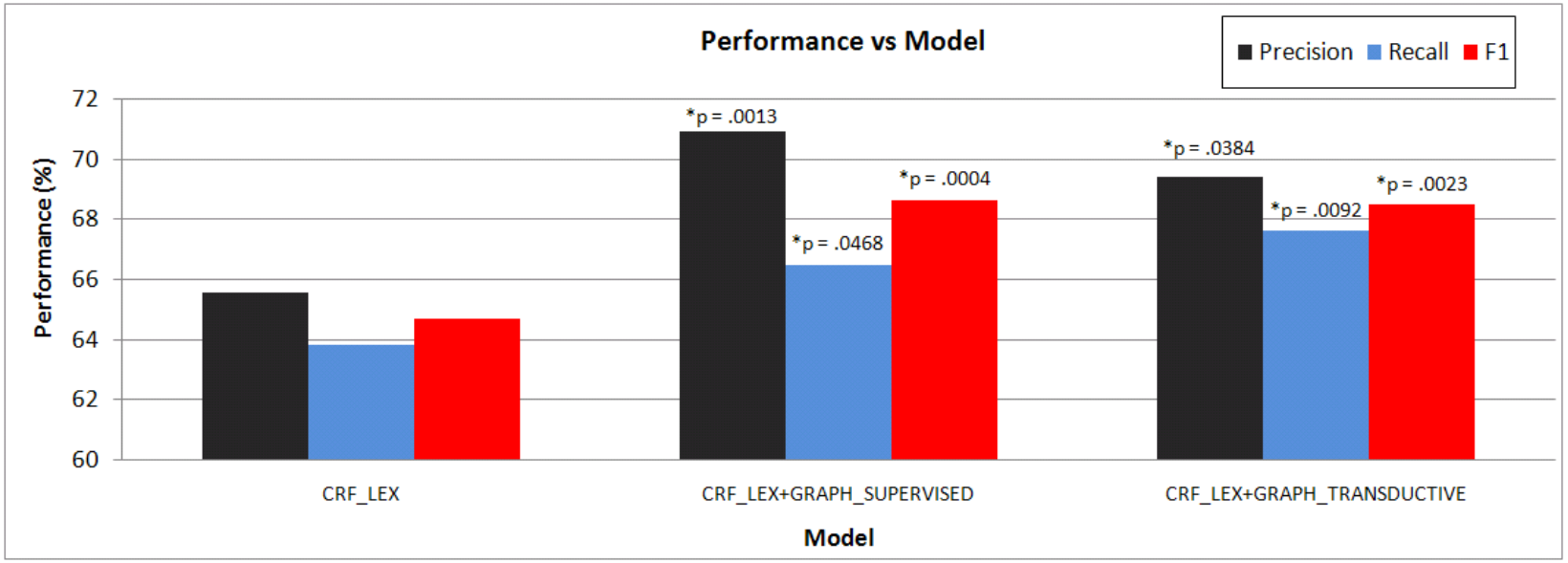


Figure 6.6: Precision (black), recall (blue), and F1 (red) of a lexical CRF model (CRF_LEX), a lexical CRF model augmented with supervised graph-based features (CRF_LEX + GRAPH_SUPERVISED), and a lexical CRF model augmented with semi-supervised graph-based features (CRF_LEX+GRAPH_TRANSDUCTIVE). *'s represent values which are significantly greater than the CRF model's respective value, as measured with the Wilcoxon signed rank test at the significance level (**p**) shown.

This may be because the transductive test data available at train time is unsupervised, and therefore, contains no examples of test-domain proteins or links between those proteins and author or paper nodes. Instead, all that is provided is the citation network of the test-domain data.

Under ideal circumstances, even this unlabeled citation information might help in the transductive setting by allowing probability mass to flow through 'short cuts' present in the test-domain data, but not in the training domain. For instance, if the training data is drawn from a certain journal, and the test (and therefore, transductive) data is drawn from a different journal, it may be possible that the most relevant gene for a particular author query happens to have been written about, in a training data journal, by a coauthor with whom the query author's only collaboration occurred in the *transductive domain* journal – a relationship that could not be found by the purely supervised method. This type of network would allow a transductively trained extractor to weight such test-domain relevant *paths* more highly than the purely training-domain trained counterpart, and thus provide more robust performance on cross-domain tasks.

The fact that, in our experiments, the transductive data does *not* improve performance seems to indicate that the form of the training and test networks in our data, along with the paths contained within them, are relatively consistent across domains, and that the training data set is sufficiently large. Although we did not perform the experiments here, it would be interesting to examine this relationship between the amount of content-based information (such as lexical features) available in the data to the usefulness of network-based information (such as the citation features). Specifically, by varying the amount of lexical information provided to the algorithm at train time, one could help tease out how the lexical, supervised, and transductive graph-based features relate to each other and in what situations each type of data (content vs. network) is most useful.

6.2.6 Conclusions

In this section we have shown that we can use *source-domain* abstracts labeled with *source-task* lexical features, along with *auxiliary-domain* paper_ids labeled with *auxiliary-task* authors, citations and genes to help identify unlabeled genes. This is an example of **domain adaptive multi-task transfer**, since it suggests that learning to predict which proteins an author is likely to write about will also help recognizing those proteins in text. Similar techniques could be used to exploit recognition of other auxiliary information in the text, such as the author or subject organism.

We have also shown that graphs, such as the curated citation network in §6.1, provide a convenient structure with which to relate and integrate **external data sources** such as SGD and GO. We have shown that random walks over these graphs can be used to generate **soft labels**, or priors, which can then be used as features to significantly improve standard information extraction techniques for gene named entity extraction. This method allows for clean and efficient **incorporation of disparate sources of information** derived from external data sources into existing information extraction systems, aiding not just in learning but also at evaluation time. This, in turn, leads to the identification and exploitation of stable relationships that contribute to robust learning in transfer settings.

Chapter 7

Conclusion

7.1 Summary

In Chapter 1 of this thesis we introduced and motivated the study of robust named entity recognition, continuing on to explore the landscape of existing techniques and settings in Chapter 2. We have shown a number of ways by which regularities and relationships in the instance, feature and label space of domains and tasks can be exploited to improve the robustness of the trained learners. Specifically, as summarized in Figure 1.1, we have:

- used the linguistically-inspired **feature hierarchies** of Chapter 3 to exploit the hierarchical relationship between lexical features, allowing for natural smoothing and sharing of information across features.

- developed the **structural frequency features** of Chapter 4 to take advantage of the information contained in the very structure of the data itself and the distribution of instances across that structure (in our case, the distribution of words across the sections of an article).

- introduced **snippets** in Chapter 5, leveraging the relationship of entities among themselves, across tasks and labels within a dataset. Specifically, we used high confidence positive and negative labels of various kinds to pseudo-label unannotated target domain data, thus allowing us to learn a more robust representation of the entire space of data.

- explored the simple, yet powerful, graph relations of Chapter 6 which allow us to tie together disparate entities and sources of data in a way that lets brittleness in one cross-domain task be supported by robustness from another, orthogonal, task.

7.2 Overview: generalizability & extensions

While these specific techniques for achieving robust named entity recognition in a transfer setting have proven very successful and constitute a substantial part of this thesis' contribution, we would now like to take a step back and present a higher-level view of what general principles we can distill from our results that might prove generally applicable to other machine learning practitioners in potentially quite different domains.

While others have presented 'unifying' views of transfer and regularization with respect to hierarchical Bayesian models [Finkel and Manning, 2009] and spectral graph based methods [Dai et al., 2009], we view the various approaches discussed in the previous chapters as all being characterized as methods for leveraging different types of *metadata* that may be shared across domains and tasks. As first introduced in Chapter 6, we use the term *metadata* to refer to aspects of data and tasks that are distinct from the domain and task-specific instance features or class labels commonly used in machine learning problems. In the visualization of Figure 1.1, these are the lines and clouds that join problems together (Z) even when their feature or label spaces differ (X and Y).

Given this perspective, we can then reinterpret each of our robust learning techniques as a method for isolating and exploiting different forms of metadata. For example, tokens occurring in different sections of a document share meta properties such as: their linguistic features having common parents (exploited by lexical *feature hierarchies* to propagate feature weights); belonging to a common *discourse* (exploited by *snippets* to propagate class labels); being contained in an integrated *document* (exploited by *structural frequency features* to compute section-invariant features); and having been written by a single set of *authors* (exploited by *citation networks* to propagate other shared features, such as *mentioned genes*). In this way, transfer is achieved by allowing some information to propagate from source to target, conducted along paths of shared metadata.

We have further found that combining these metadata can have complimentary effects: stringing together features so as to construct a chain or network of interconnected properties (as in our *citation networks*) allows transfer among domains that may have no direct connection between them (such as discovering sets of authors who have never written a paper together yet share common interests).

Metadata, of course, is not limited to the text-based NER problems discussed in this thesis. Indeed, the common definition of transfer learning as 'training and testing on different, but related, domains or tasks' presupposes the source and target problems to be 'related.' It is exactly these shared *relationships*, linking the two learning problems together, that a practitioner can exploit as metadata. The key is to distill out exactly what these common properties are and how they are distributed across the domains adn tasks: whether hierarchically as in our *lexical features*, identically as in our *snippets*, or in more complex patterns as in our *citation networks*. Once these relationships are understood, they can be represented explicitly as features and fed into any common learning algorithm[1].

[1]While we focus on conditional random fields and random walks in this work, there is no reason these metadata-based features could not be used by any other machine learning algorithm.

As an example of the generality of this concept of shared metadata, consider the problem of categorizing the sentiment of blog posts on the internet. We may believe that training on one set of blogs (say, technology news sites) should help us perform better on a different set of blogs (say, product reviews), but we may not be exactly sure how to transfer knowledge from one domain to the other. Using our approach, we would first identify metadata that is shared among various aspects of the domains and tasks. For example, if we are able to perform product name recognition in the news domain, we may be able to identify common products that appear both in news and product review sites by matching tokens classified as products in the news sites to their counterparts in the review sites. We can then use this identity property to link the associated posts together, in a way analogous to *snippets*. Similarly, we can exploit the blog publication process by grouping posts written by common authors, or published on common sites, together, as in our *citation networks*. We can go further and examine the structure of a blog post itself: following hyperlinks, parsing comments (including attributing them to their authors) and linking tracebacks. For instance, we can apply the one-sense-per-discourse assumption to the comments on a blog post to conclude that if a comment is recognized to be **a)** strongly disagreeing with the parent post and **b)** clearly expressing negative sentiment, then the original poster was most likely expressing positive sentiment.

Another example is the problem of identifying product names in company print brochures given crawls of other companies' websites. In this case, there are essentially two types of transfer going on at once: from website to brochure, and from one company to another. Again, the first step is to identify common attributes shared across the different web, print, and company domains. For example, in terms of transferring product names across companies, we might be able to identify key words on company websites that signal what industry the company is in. Using these as domain-invariant features, we can construct industry-specific grammars for indentifying product names: for example, software products

may commonly end in a version number such as '2.0', while car products might end in a series code such as 'EX'. If we have access to external data sources, such as resume websites like `LinkedIn`, we might be able to link companies together in some way by the work histories of their employees. Likewise, shared formatting might provide a means of transferring from web to print, leveraging the assumption, for example, that usage of the italics tag in HTML is analogous to usage of italics font in print.

We hope these examples have helped demonstrate how the concept of relating shared metadata across domains and tasks can provide a general framework for performing robust transfer in a wide variety of different learning settings and encourage other practitioners to take a look at their own problems and consider if a metadata approach might be beneficial.

7.3 Future work

There remain many rich avenues of exploration in this topic of robust learning for named entity recognition. For example, while we used the seemingly natural tree-like structure of the MinorThird feature-space to construct our **feature hierarchy**, there are many more ways one could think about creating, or learning, this structure – whether from prior domain knowledge or a model selection algorithm.

For instance, one could imagine directly assessing the correlation between various sets of features in some validation dataset, and using this information to help reorder the feature tree. Specifically, different versions of the tree could be constructed by 'pivoting' the tree on various **key** features (as determined by these features' covariances and correlations with other features and statistics), and the tress created in this manner compared in a cross-validation model selection experminent.

In fact, the idea of organizing and relating entities together via hierarchies is not limited to

features. Whereas the **snippets** we examined in this thesis relied on exact token matching to perform their function, one could imagine a hierarchical system that allowed one to robustly back-off from *exact* token matches to *tree-based* similarity matches. Such a system might be able to relate a specific species, such as '*Homo sapiens*', to other related cousins, such as '*Homo erectus*', merely by recognizing their common subtoken '*Homo*'. This could be encoded as a common *parent* token node '*Homo*', having '*sapiens*' and '*erectus*' as sibling child nodes. In this way a purely token-based hierarchy might be able to capture seemingly more robust semantic information rather cheaply.

Similarly, one could search out many other types of structure within and around documents to exploit along the lines of the document sections used by our **structural frequency features**. In the biological domain, for example, papers can often be categorized by the procedure or experimental method employed. It seems promising that there would be certain regularities within and between experiment types that might be useful for identifying proteins and other associated discriminative words.

With respect to **snippets**, the continued development of task-specific classifiers, such as image pointers, provides a reusable, modular type-system on top of which robust learners of many kinds can be built. Relatedly, the development of taxonomies and grammars that allow domain experts to encode the relationships and constraints between different entities and aspects of the data promises to be a challenging, but rewarding, area of future work.

Tying all of these approaches together is the framework of using graphs to encode and operate on relationships between data of all types. Our feature hierarchies are just trees (a special type of graph) relating features to one another, while snippets and structural frequency features both encode similarity and distinction between tokens across different aspects of the data and domains being learned. These links, joining information across feature-, instance- and label-space, can likewise be modeled as edges in a complex network. Such graphical

models, and their associated walks, edge-learning and other techniques, *themselves* provide a robust method by which to combine data from multiple sources in order to discover regularities that lead, ultimately, to learning that is more robust than any single technique alone.

With the exception of these graph-walks, we have focused, for ease of development and comparison, on a relatively small set of linear learning methods such as conditional random fields. While we feel that these techniques healthily served, and did not limit, our research, there is certainly room to benefit from further extension and incorporation of our methods to other machine learning algorithms such as latent variable or grammar based models.

With respect to **curated citation networks**, it is still unclear exactly how much information can be combined before diminishing returns are encountered. For instance, we know that adding a single new query significantly improves prediction accuracy, but what about adding a second or third gene? Relatedly, does it matter if the gene provided is relatively common or rare in the over distribution of genes across publications? Does knowledge of a harder to recognize gene provide more benefit that an easy one?

Following up on an existing model of document structure [Cohen et al., 2003; Arnold and Cohen, 2008], we hope to extend this annotated citation network model to include the text and structure of the document itself. In this proposed model, different sections of a document, such as the abstract and image captions, along with the words that occur in those sections, would be represented as nodes in a document graph, with edges connecting related entities (such as the sequential ordering of paragraphs or cross-referencing of images and citations in the text).

We also see value in incorporating a temporal dimension to the network. In our current model all edges are walked upon with equal probability, regardless of the temporal distance between the two connected nodes. We might do better by taking this time distance into

account: for example, coauthorship on a paper 20 years ago may carry less weight than a collaboration just a few years ago.

In addition to leveraging traditional information extraction techniques to aid in our graph search, as already demonstrated, we would also like to explicitly model the text of the document as a graph [Minkov et al., 2006]. For example, yeast gene names share morphological features: all genes sharing the same three letter prefix are functionally related. Likewise, we hope to deepen the gene-gene relationships we extracted crudely from parsing the textual GO annotation data. One way of doing this would be to calculate the mutual information of pairs of genes by measuring how often they occur across different sections of a single document, or across publications related by the same author, publication year, etc. This would also provide real-valued (as opposed to binary-valued) data from which to construct weighted edges. In this way we hope to improve performance by modeling these types of relationships and information graphically.

We would also like to perform the full graph-based transfer experiment alluded to in the introduction to Chapter 6, in which we apply our graph-based features, successful in improving purely lexically-based intra-domain NER, to the cross-domain problem of identifying genes in captions, having only been trained on abstracts and their associated annotated citation networks. We could also extend this to transfer across biological subdomains, such as organisms, journals, and experiment design.

Finally, we look forward to extending the ideas begun in this thesis (summarized in Figure 1.1) for representing and integrating the various heuristics we have developed for robust learning into a common, unified framework. In particular, we see potential in expressing the feature hierarchies, snippets, structural frequency features and various citation-based information (along with other techniques and sources of data yet to be developed) as relations in a large graph which serve as 'bridges' conducting information between the source and

target domains.

Appendix A

Feature Language Definition

The lexical features used in this work are produced by the open-source natural language toolkit MinorThird [Cohen, 2004]. The user can define a method of tokenization as a regular expression (the default is to split on non-alphanumeric characters), and then the toolkit evaluates a set of predefined binary features on each token:

- *charType*: These features describe (in a regular expression-like syntax) the pattern of the characters in the token.
  - *charType.Xx+*: the token begins with a capital letter, followed by lower case letters
  - *charType.X+*: the token contains all capital letters
  - *charType.x+*: the token contains all lower case letters
  - *charType.0+*: the token contains all digits
- *isWord*: These features describe the exact value of the token. They can be modified by the *lowerCase* operator to match lower case versions of the word.

- *isWord.aardvark*
- *isWord.ant*
- *...*
- *isWord.zebra*

Each feature is recursively defined and evaluated (as **TRUE** or **FALSE**) on each token in the document, along with each token's neighbors (within a defined window size, by default three) to the left and right, as shown in Listing A.1.

In addition to these predefined features, the user is allowed to define her own set of features that will likewise be evaluated across the tokens of the document. The templates for these features can be expressed in a regular expression-like language called *Mixup* which allows great flexibility for creating customizable and extensible domain-specific feature languages. For instance, Table 3.1 demonstrates a feature called *LeftToken.1.isTitle*, that evaluates **TRUE** for the token 'Professor', one to the left of the token 'Caldwell'. To create this feature *isTitle*, the user *a priori* defined a Mixup program with a *dictionary* called 'titles' that contained the strings the user wanted to recognize as titles, such as: 'Mr.', 'Mrs.', 'Dr.', 'Professor'. Then, at training time, the Mixup program is run over the document, evaluating the *isTitle* feature (along with all the other features) over each of the tokens, evaluating as **TRUE** for the token 'Professor'.

The features in a Mixup program can also refer to one another, allowing the user to define a context-sensitive sequence of word types specifying, for instance, that the part of a token followed by the common protein suffix 'ine' should be tagged with the feature *isProteinBase*. These custom made features are applied in the same recursive hierarchical manner as the built-in features.

Listing A.1: MinorThird feature generation code.

```
curTok = currentToken;
// charType features
from(curTok).tokens().eq().charType().emit();
// isWord features
from(curTok).tokens().eq().isWord().emit();
// userDefined features
for (int j=0; userDefinedFeatures.length; j++) {
    from(curTok).tokens().prop(userDefinedFeatures[j]).emit();
}
// neighborhood features
for (int i=0; i<windowSize; i++) {
    // charType features
    from(curTok).left().token(-i-1).eq().charType().emit();
    from(curTok).right().token(i).eq().charType().emit();
    // isWord features
    from(curTok).left().token(-i-1).eq().isWord().emit();
    from(curTok).right().token(i).eq().isWord().emit();
    // userDefined features
    for (int j=0; userDefinedFeatures.length; j++) {
        from(curTok).left().token(-i-1).prop(userDefinedFeatures[j]).emit();
        from(curTok).right().token(i).prop(userDefinedFeatures[j]).emit();
    }
}
```

In these various ways the user is able to specify a robust customizable and precise definition of the features he would like to instantiate over the data, ultimately serving as the representation over which learning is performed.

Appendix B

Hierarchical Feature Model Evaluations

Figures B.1, B.2, B.3 and B.4 each show the full set of evaluations performed for each of the models described in Section 3.2.2 under each experimental setting of training and tuning data, as evaluated on MUC6, MUC7, UTexas and Yapex test data respectively.

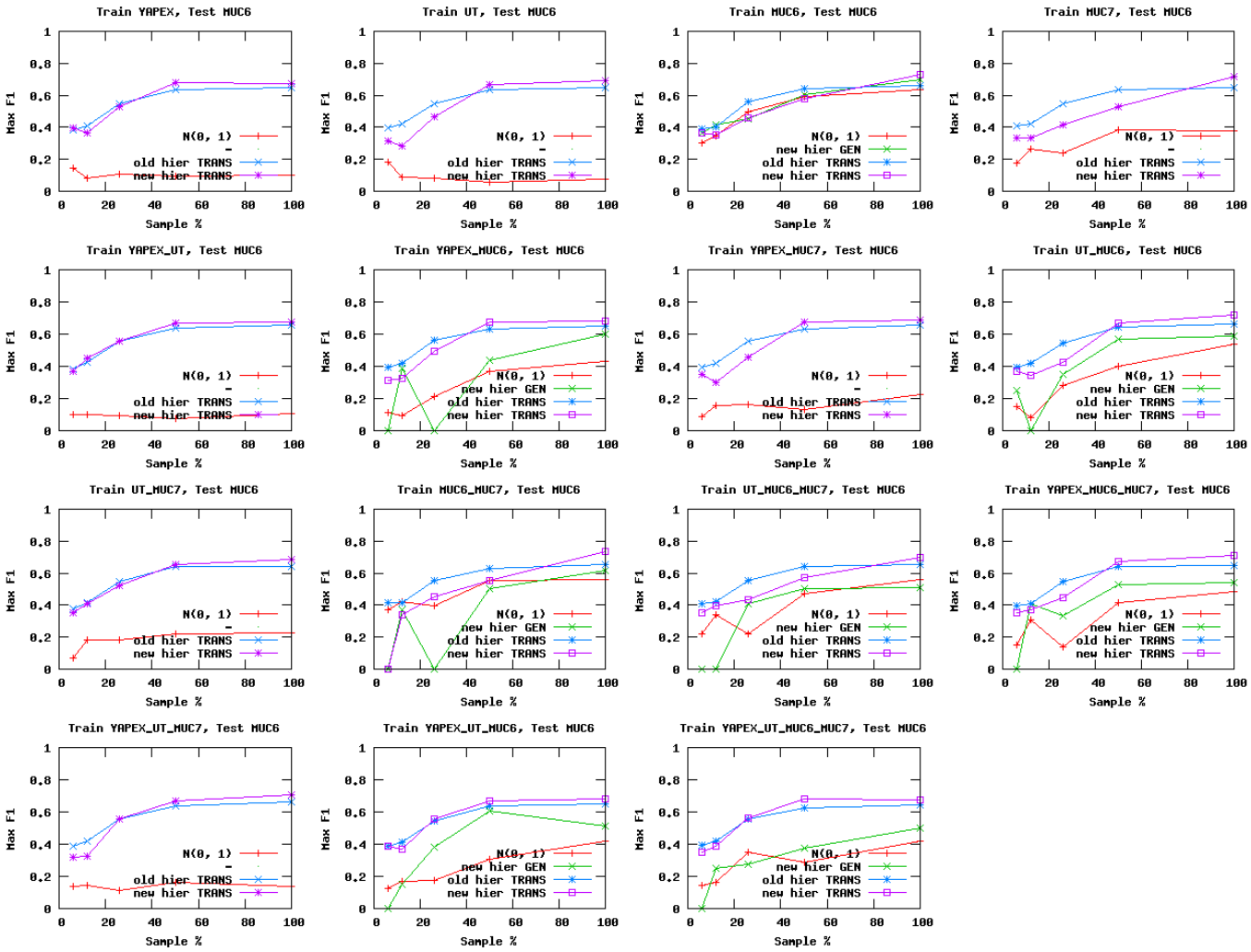


Figure B.1: Comparative results for various experiment settings evaluated on the MUC6 dataset. (**Red 'N(0,1)'** uses a standard normal regularizer, and concatenates the training data where applicable. When the train dataset is the same as the test dataset this is the GAUSS model; **Green 'new hier GEN'** uses a generalizing hierarchical model, without transfer, and so is only applicable when the target domain data is part of the training set; **Blue 'old hier TRANS'** uses our hierarchical model; **Purple 'new hier TRANS'** uses the CHELBA-ACERO model)

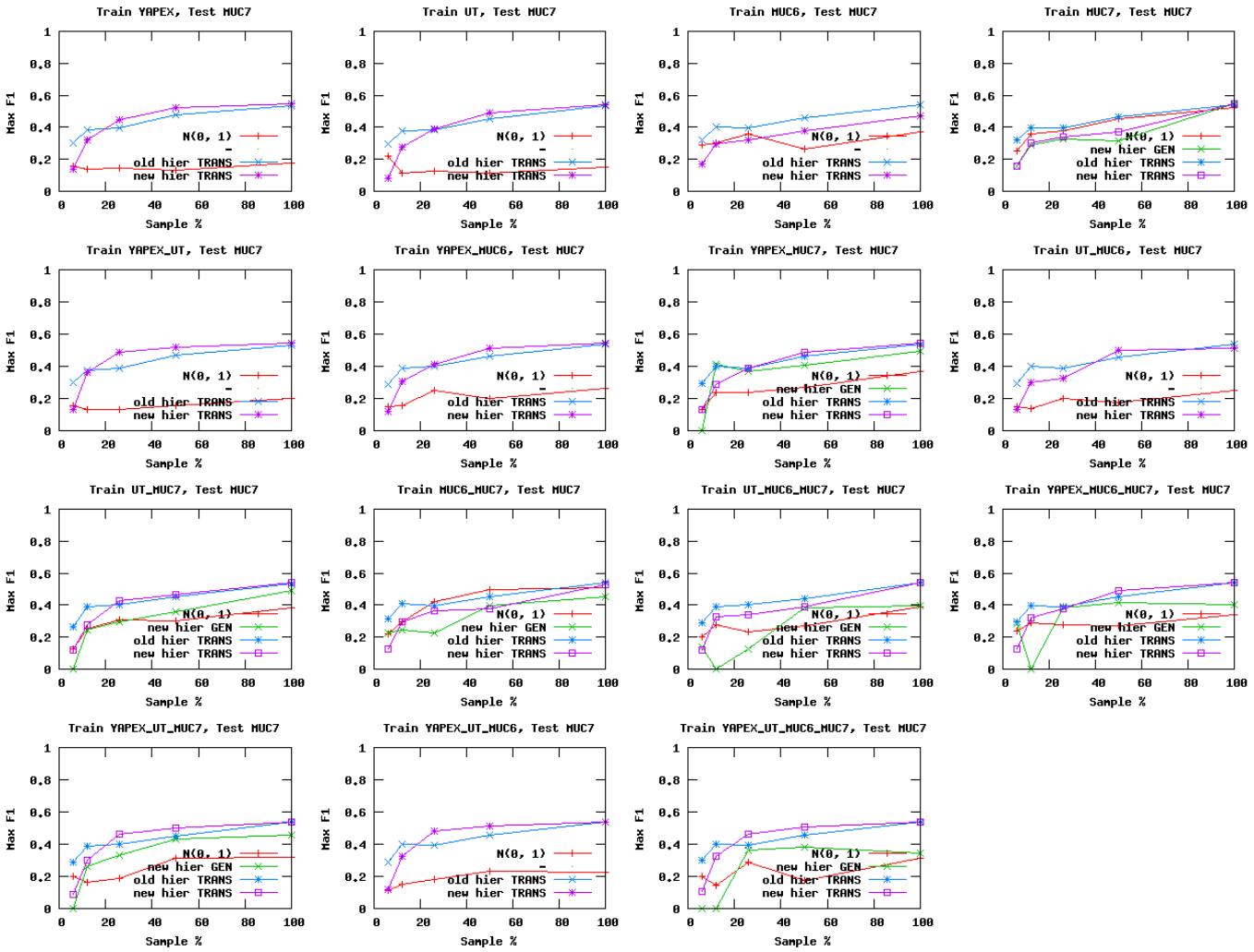


Figure B.2: Comparative results for various experiment settings evaluated on the MUC7 dataset. (**Red 'N(0,1)'** uses a standard normal regularizer, and concatenates the training data where applicable. When the train dataset is the same as the test dataset this is the GAUSS model; **Green 'new hier GEN'** uses a generalizing hierarchical model, without transfer, and so is only applicable when the target domain data is part of the training set; **Blue 'old hier TRANS'** uses our hierarchical model; **Purple 'new hier TRANS'** uses the CHELBA-ACERO model)

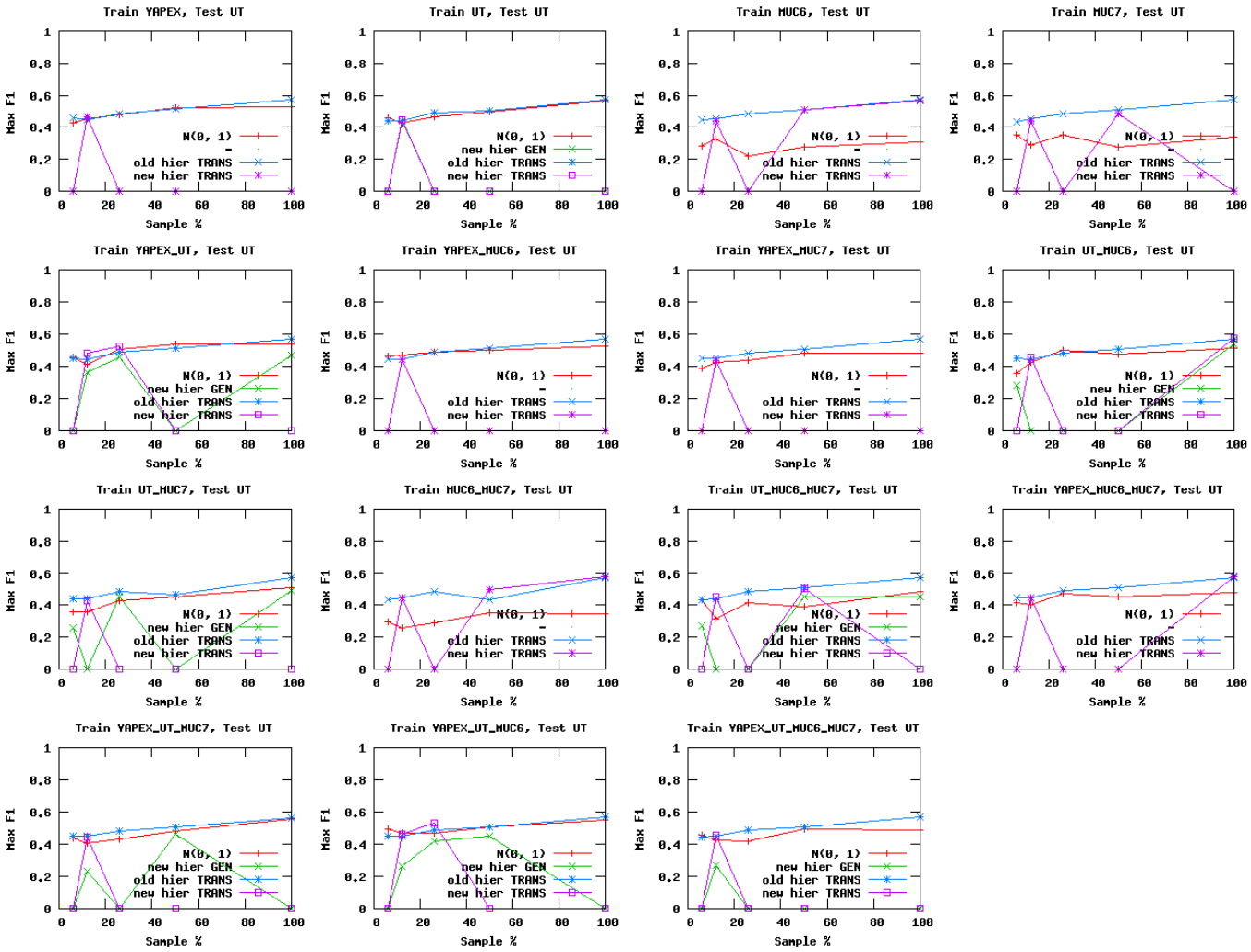


Figure B.3: Comparative results for various experiment settings evaluated on the UTexas dataset. (**Red 'N(0,1)'** uses a standard normal regularizer, and concatenates the training data where applicable. When the train dataset is the same as the test dataset this is the GAUSS model; **Green 'new hier GEN'** uses a generalizing hierarchical model, without transfer, and so is only applicable when the target domain data is part of the training set; **Blue 'old hier TRANS'** uses our hierarchical model; **Purple 'new hier TRANS'** uses the CHELBA-ACERO model)

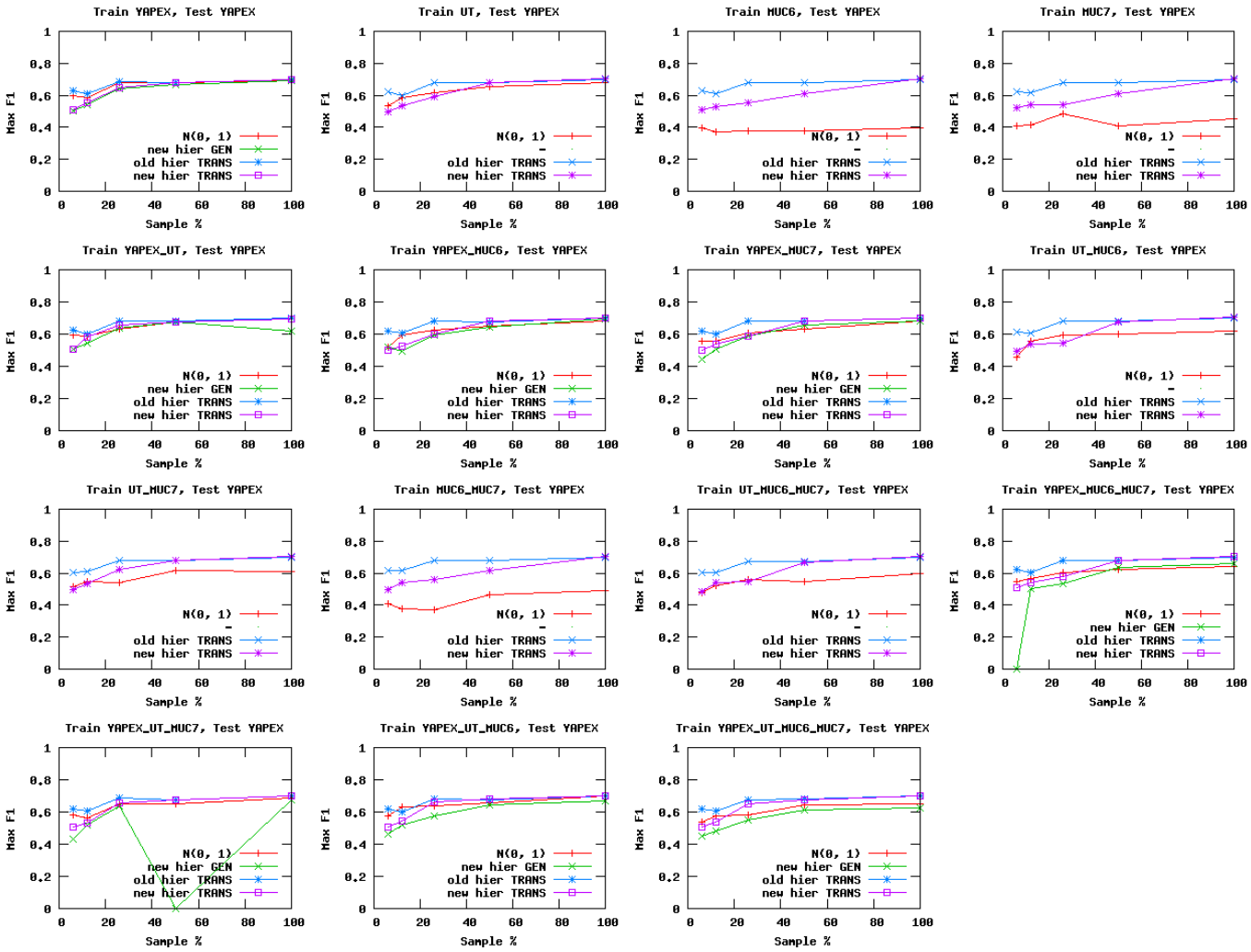


Figure B.4: Comparative results for various experiment settings evaluated on the Yapex dataset. (**Red 'N(0,1)'** uses a standard normal regularizer, and concatenates the training data where applicable. When the train dataset is the same as the test dataset this is the GAUSS model; **Green 'new hier GEN'** uses a generalizing hierarchical model, without transfer, and so is only applicable when the target domain data is part of the training set; **Blue 'old hier TRANS'** uses our hierarchical model; **Purple 'new hier TRANS'** uses the CHELBA-ACERO model)

Bibliography

[Abney, 2007] Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. Chapman & Hall.

[Ando and Zhang, 2005] Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. In *JMLR 6*, pages 1817 – 1853.

[Arnold and Cohen, 2008] Arnold, A. and Cohen, W. W. (2008). Intra-document structural frequency features for semi-supervised domain adaptation. In *CIKM '08*.

[Arnold and Cohen, 2009] Arnold, A. and Cohen, W. W. (2009). Information extraction as link prediction: Using curated citation networks to improve gene detection. In *ICWSM '09*.

[Arnold et al., 2007] Arnold, A., Nallapati, R., and Cohen, W. W. (2007). A comparative study of methods for transductive transfer learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM) 2007 Workshop on Mining and Management of Biological Data*.

[Arnold et al., 2008] Arnold, A., Nallapati, R., and Cohen, W. W. (2008). Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL:HLT '08*.

[Baxter, 1997] Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39.

[Ben-David et al., 2007] Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *NIPS 20*, Cambridge, MA. MIT Press.

[Berger et al., 1996] Berger, A. L., Pietra, S. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

[Blei et al., 2002] Blei, D. M., Bagnell, J. A., and McCallum, A. (2002). Learning with scope, with application to information extraction and classification. In *UAI*, pages 53–60.

[Blitzer et al., 2006] Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *EMNLP*, Sydney, Australia.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 92–100.

[Borthwick et al., 1998] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. (1998). NYU: Description of the MENE named entity system as used in MUC-7.

[Bunescu et al., 2004] Bunescu, R., Ge, R., Kate, R., Marcotte, E., Mooney, R., Ramani, A., and Wong, Y. (2004). Comparative experiments on learning information extractors for proteins and their interactions. In *Journal of AI in Medicine*. Data from *ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/proteins.tar.gz*.

[Cardillo et al., 2006] Cardillo, A., Scellato, S., and Latora, V. (2006). A topological analysis of scientific coauthorship networks. In *Physica A: Statistical Mechanics and its Applications*.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

[Chelba and Acero, 2004] Chelba, C. and Acero, A. (2004). Adaptation of maximum entropy capitalizer: Little data can help a lot. In Lin, D. and Wu, D., editors, *EMNLP 2004*, pages 285–292. ACL.

[Chen and Rosenfeld, 1999] Chen, S. and Rosenfeld, R. (1999). A gaussian prior for smoothing maximum entropy models.

[Cohen and Hersh, 2005] Cohen, A. M. and Hersh, W. R. (2005). A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6:57–71.

[Cohen, 2004] Cohen, W. W. (2004). Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net.

[Cohen and Carvalho, 2005] Cohen, W. W. and Carvalho, V. (2005). Stacked sequential learning. *International Joint Conferences on Artificial Intelligence (IJCAI)*.

[Cohen and Minkov, 2006] Cohen, W. W. and Minkov, E. (2006). A graph-search framework for associating gene identifiers with documents. *BMC Bioinformatics*, 7(440).

[Cohen et al., 2003] Cohen, W. W., Wang, R., and Murphy, R. (2003). Understanding captions in biomedical publications. In *KDD*, pages 499–504.

[Cohn and Hofmann, 2001] Cohn, D. and Hofmann, T. (2001). The missing link: A probabilistic model of document content and hypertext connectivity. In *NIPS*.

[Collins and Singer, 1999] Collins, M. and Singer, Y. (1999). Unsupervised models for named entity classification. In *Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

[Consortium, 2000] Consortium, T. G. O. (2000). Gene ontology: tool for the unification of biology. In *Nature Genet*, volume 25, pages 25–29.

[Dai et al., 2009] Dai, W., Jin, O., Xue, G.-R., Yang, Q., and Yu, Y. (2009). Eigentransfer: a unified framework for transfer learning. In *ICML*.

[Daumé III, 2007] Daumé III, H. (2007). Frustratingly easy domain adaptation. In *ACL*.

[Daumé III, 2008] Daumé III, H. (2008). Cross-task knowledge-constrained self training. In *Proceedings of EMNLP*.

[Daumé III and Marcu, 2006] Daumé III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research 26*, pages 101–126.

[Dempster et al., 1977] Dempster, A., Laird, N., , and Rubin, D. (1977). Likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society, Series B*.

[Dietz et al., 2007] Dietz, L., Bickel, S., and Scheffer, T. (2007). Unsupervised prediction of citation influences. In *ICML*.

[Donoho and Johnstone, 1995] Donoho, D. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224.

[Dwight et al., 2004] Dwight, S. S., Balakrishnan, R., Christie, K. R., Costanzo, M. C., Dolinski, K., Engel, S. R., Feierbach, B., Fisk, D. G., Hirschman, J., Hong, E. L., Issel-Tarver, L., Nash, R. S., Sethuraman, A., Starr, B., Theesfeld, C. L., Andrada, R., Binkley, G., Dong, Q., Lane, C., Schroeder, M., Weng, S., Botstein, D., and Cherry, J. M. (2004). Saccharomyces genome database: underlying principles and organisation. *Brief Bioinform.*, 5(1):9–22. ftp://ftp.yeastgenome.org/yeast/.

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. In *Annals of Statistics*.

[Erosheva et al., 2004] Erosheva, E., Fienberg, S., and Lafferty, J. (2004). Mixed membership models of scientific publications. *PNAS*, 101(21).

[Feldman et al., 2003] Feldman, R., Regev, Y., Finkelstein-Landau, M., Hurvitz, E., and Kogan, B. (2003). Mining the biomedical literature using semantic analysis. *Biosilico*, 1(2):69–80.

[Finkel and Manning, 2009] Finkel, J. R. and Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *NAACL*.

[Fisher et al., 1995] Fisher, D., Soderland, S., McCarthy, J., Feng, F., and Lehnert, W. (1995). Description of the UMass system as used for MUC-6.

[Franzén et al., 2002] Franzén, K., Eriksson, G., Olsson, F., Asker, L., Lidn, P., and Cöster, J. (2002). Protein names and how to find them. In *International Journal of Medical Informatics*.

[Gale et al., 1992] Gale, W. A., Church, K. W., and Yarowsky, D. (1992). One sense per discourse. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 233–237, Morristown, NJ, USA. Association for Computational Linguistics.

[Garfield et al., 1964] Garfield, E., Sher, I., and Torpie, R. (1964). *The Use of Citation Data in Writing the History of Science*. The Institute for Scientific Information.

[Ghahramami and Jordan, 1994] Ghahramami, Z. and Jordan, M. (1994). Learning from incomplete data. In *Technical Report, AI Lab No. 1509*.

[Ghamrawi and McCallum, 2005] Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *CIKM*.

[Globerson and Roweis, 2006] Globerson, A. and Roweis, S. T. (2006). Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*.

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):6170.

[Grandvalet and Bengio, 2005] Grandvalet, Y. and Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In *CAP*, Nice, France.

[Janche and Abney, 2002] Janche, M. and Abney, S. P. (2002). Information extraction from voicemail transcripts. In *EMNLP*.

[Ji et al., 2002] Ji, K., Ohta, M., and Tsujii, Y. (2002). Tuning support vector machines for biomedical named entity recognition. In *ACL Workshop on Natural Language Processing in the Biomedical Domain*.

[Jiang and Zhai, 2006] Jiang, J. and Zhai, C. (2006). Exploiting domain structure for named entity recognition. In *Human Language Technology Conference*, pages 74 – 81.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *ICML 16*.

[Joachims, 2002] Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines*. Kluwer.

[Joachims, 2003] Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *ICML*.

[Jolliffe, 2002] Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. In *JACM*.

[Kleinberg et al., 1999] Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The web as a graph: Measurements, models and methods. In *Lecture Notes in Computer Science*.

[Kou and Cohen, 2007] Kou, Z. and Cohen, W. W. (2007). Stacked graphical models for effcient inference in markov random fields. *SIAM Data Mining Conference (SDM)*.

[Kraut et al., 2004] Kraut, R., Fussell, S., Lerch, F., and Espinosa, J. (2004). Coordination in teams: evidence from a simulated management game.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

[Lee et al., 2007] Lee, S.-I., Chatalbashev, V., Vickrey, D., and Koller, D. (2007). Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of International Conference on Machine Learning (ICML)*.

[Leicht et al., 2007] Leicht, E. A., Clarkson, G., Shedden, K., and Newman, M. E. J. (2007). Large-scale structure of time evolving citation networks. *Eur. Phys. J. B*, 59:75–83.

[Liben-Nowell and Kleinberg., 2003] Liben-Nowell, D. and Kleinberg., J. (2003). The link prediction problem for social networks. In *CIKM*.

[Liu et al., 2005] Liu, X., Bollen, J., Nelson, M., and de Sompel, H. V. (2005). Co-authorship networks in the digital library research community. In *Information Processing and Management,*.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *In AAAI Workshop on Learning for Text Categorization*.

[Minkov et al., 2006] Minkov, E., Cohen, W. W., and Ng, A. Y. (2006). Contextual search and name disambiguation in email using graphs. In *SIGIR*.

[Minkov et al., 2005] Minkov, E., Wang, R. C., and Cohen, W. W. (2005). Extracting personal names from email: Applying named entity recognition to informal text. In *HLT/EMNLP*.

[Murphy et al., 2004] Murphy, R. F., Kou, Z., Hua, J., Joffe, M., and Cohen, W. W. (2004). Extracting and structuring subcellular location information from on-line journal articles: The subcellular location image finder. In *KSCE*.

[National Institues of Health, 2009] National Institues of Health (2009). http://www.pubmedcentral.nih.gov/.

[Nigam et al., 1999] Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification.

[Nigam et al., 2000] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

[Ohta et al., 2002] Ohta, T., Tateisi, Y., Mima, H., and Tsujii, J. (2002). Genia corpus: an annotated research abstract corpus in molecular biology domain. In *HLT: Human Language Technology Conference*, pages 92–100.

[Raina et al., 2006] Raina, R., Ng, A. Y., and Koller, D. (2006). Transfer learning by constructing informative priors. In *ICML 22*.

[Schölkopf et al., 2005] Schölkopf, B., Steinke, F., and Blanz, V. (2005). Object correspondence as a machine learning problem. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 776–783, New York, NY, USA. ACM.

[Shi and Campagne, 2005] Shi, L. and Campagne, F. (2005). Building a protein name dictionary from full text: a machine learning term extraction approach. *BMC Bioinformatics*, 6(88).

[Sindhwani et al., 2005] Sindhwani, V., Niyogi, P., and Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, pages 824–831. ACM.

[Sutton and McCallum, 2004] Sutton, C. and McCallum, A. (2004). Collective segmentation and labeling of distant entities in information extraction. *ICML workshop on Statistical Relational Learning*.

[Sutton and McCallum, 2005] Sutton, C. and McCallum, A. (2005). Composition of conditional random fields for transfer learning. In *HLT/EMLNLP*.

[Szafranski et al., 2007] Szafranski, M., Grandvalet, Y., and Morizet-Mahoudeaux, P. (2007). Hierarchical penalization. In *Advances in Neural Information Processing Systems 20*. MIT press.

[Taskar et al., 2003] Taskar, B., Wong, M.-F., and Koller, D. (2003). Learning on the test data: Leveraging 'unseen' features. In *Proc. Twentieth International Conference on Machine Learning (ICML)*.

[Thrun, 1996] Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *NIPS*, volume 8, pages 640–646. MIT.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. In *J. Royal. Statist. Soc B*.

[U.S. National Library of Medicine, 2008] U.S. National Library of Medicine (2008). http://www.ncbi.nlm.nih.gov/pubmed/.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Wang et al., 2008] Wang, R. C., Tomasic, A., Frederking, R. E., Simmons, I., and Cohen, W. W. (2008). Learning to extract gene-protein names from weakly-labeled text. *CMU SCS Technical Report Series*, CMU-LTI-08-004.

[Widmer and Kubat, 1996] Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. In *Machine Learning*.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[Yang, 2001] Yang, Y. (2001). A study of thresholding strategies for text categorization. In *SIGIR*.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*.

[Zaffalon and Hutter, 2002] Zaffalon, M. and Hutter, M. (2002). Robust feature selection by mutual information distributions. In *18th Conference on Uncertainty in Artificial Intelligence*.

[Zhang et al., 2005] Zhang, J., Ghahramani, Z., and Yang, Y. (2005). Learning multiple related tasks using latent independent component analysis.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. In *Technical Report 1530*. University of Wisconsin.

[Zhu and Ghahramani, 2002] Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.